ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
Τμήμα Μηχ/κών Η/Υ, Τηλεπικοινωνιών & Δικτύων

Διάδοση Εμπιστοσύνης με μεθόδους βασισμένες στη Φυσική

ΤΥΧΟΓΙΩΡΓΟΣ ΓΙΩΡΓΟΣ

ΠΡΟΠΤΥΧΙΑΚΟΣ ΦΟΙΤΗΤΗΣ ΤΜΗΜΑΤΟΣ
ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ : ΚΟΥΤΣΟΠΟΥΛΟΣ ΙΟΡΔΑΝΗΣ

ΒΟΛΟΣ 2006

Ευχαριστίες

Στην προσπάθειά μου αυτή συνέβαλαν πολλοί άνθρωποι τους οποίους νιώθω την ανάγκη να ευχαριστήσω.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου διατριβής κ. Ιορδάνη Κουτσόπουλο, γιατί ήταν πάντοτε πρόθυμος να προσφέρει κάθε δυνατή βοήθεια για την αντιμετώπιση οποιονδήποτε δυσκολιών. Επίσης, τον κ. Λέανδρο Τασσιούλα για τις πολύτιμες συμβουλές του κατά τη διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας.

Θα ήθελα, επίσης, να ευχαριστήσω όλους τους φίλους μου, και ιδιαίτερα τον Λάζαρο Γκατζίκη για τη συνεργασία μας τα πέντε αυτά χρόνια στη σχολή. Η βοήθειά του ως συμφοιτητής αλλά και ως φίλος ήταν πολύ σημαντική για μένα.

Τέλος, θεωρώ απαραίτητο να ευχαριστήσω την οικογένειά μου, που με στηρίζει πάντα σε κάθε μου επιλογή και με βοηθά να πραγματοποιήσω τα όνειρά μου.

ΠΕΡΙΕΧΟΜΕΝΑ

Σύνοψη

Τα τελευταία χρόνια παρατηρείται ταχύτατη διάδοση των **Peer to Peer (P2P)** δικτύων και η χρήση τους επεκτείνεται συνεχώς σε νέους τομείς, με πιο γνωστή την χρήση τους για διακίνηση αρχείων. Τα δίκτυα αυτά, σε αντίθεση με την καθιερωμένη αρχιτεκτονική πελάτη - εξυπηρετητή (client-server), όπου υπάρχει ένας κεντρικός εξυπηρετητής με τον οποίο επικοινωνούν όλοι οι πελάτες, αποτελούνται από υπολογιστές, που λέγονται peers και δρουν ταυτόχρονα ως πελάτες και ως εξυπηρετητές. Κάθε χρήστης, ως πελάτης παράγει αιτήσεις, με κάποιο ρυθμό, τις οποίες διαχωρίζει ("σπάει") στους υπόλοιπους εξυπηρετητές. Ως εξυπηρετητής κάθε χρήστης εξυπηρετεί τις εισερχόμενες αιτήσεις βάσει μιας πολιτικής προτεραιοτήτων.

i. Ένα βασικό κριτήριο απόδοσης των δικτύων αυτών είναι η μέση καθυστέρηση απόκτησης ενός αρχείου, η οποία εξαρτάται από τον διαχωρισμό των αιτήσεων στους κόμβους άλλα και από την πολιτική εξυπηρέτησης κάθε εξυπηρετητή. Σκοπός της εργασίας αυτής είναι η λύση του προβλήματος ελαχιστοποίησης της μέσης βεβαρημένης καθυστέρησης στο δίκτυο, όπου η καθυστέρηση κάθε χρήστη εκφράζεται είτε ως η μέση καθυστέρηση του χρήστη στο σύστημα είτε ως η το μέγιστο εκ των καθυστερήσεων σε κάποιο εξυπηρετητή. Η πρώτη περίπτωση είναι μια συνηθισμένη μετρική απόδοσης στην οποία θα χρησιμοποιήσουμε τον cμ rule για τον καθορισμό των προτεραιοτήτων από τους εξυπηρετητές. Η δεύτερη εκφράζει την περίπτωση παράλληλης εξυπηρέτησης από τους διάφορους εξυπηρετητές και μελετούμε αν ισχύει ο cμ rule. Για την εύρεση του βέλτιστου τρόπου διαχωρισμού των αιτήσεων παρουσιάζουμε έναν κατανεμημένο **αλτρουιστικό** αλγόριθμο, όπου κάθε χρήστης ελαχιστοποιεί μια έκφραση που συμπεριλαμβάνει και τις καθυστερήσεις των κόμβων μικρότερης προτεραιότητας.

Ο αλγόριθμος αυτός βασίζεται στην τεχνική του waterfilling και παρουσιάζουμε τον τρόπο με τον οποίο είναι δυνατή η υλοποίησή του με κατανεμημένο τρόπο και με ελάχιστη ανταλλαγή μηνυμάτων ανάμεσα στους κόμβους.

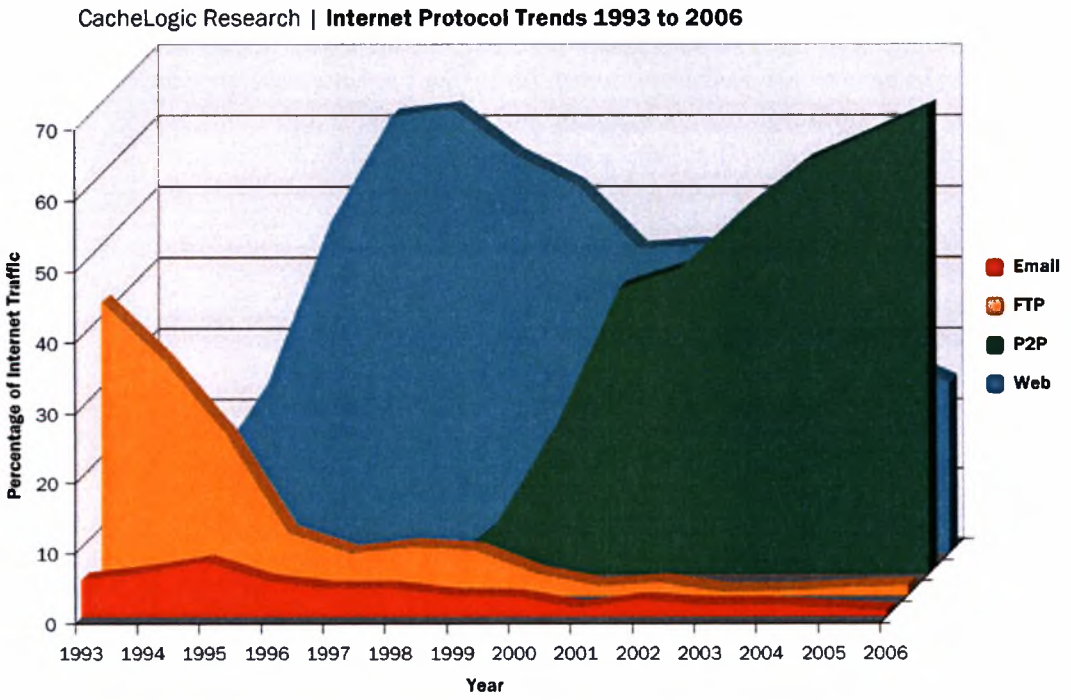
Η εργασία οργανώνεται ως εξής. Στο κεφάλαιο 1 κάνουμε μια εισαγωγή στα P2P δίκτυα παρουσιάζοντας τα κύρια χαρακτηριστικά και εφαρμογές τους. Στο κεφάλαιο 2 παρουσιάζουμε προηγούμενες εργασίες για τα δίκτυα αυτά. Στο κεφάλαιο 3 περιγράφουμε το μοντέλο του συστήματος. Τα κεφάλαια 4 και 5 περιλαμβάνουν την περιγραφή και ανάλυση των προτεινόμενων αλγορίθμων για τον υπολογισμό της καθυστέρησης ως ο μέσος όρος ή το μέγιστο της καθυστέρησης σε κάθε εξυπηρετητή αντίστοιχα . Τέλος, στο κεφάλαιο 6 αναφέρονται τα συμπεράσματα από την παρούσα εργασία.

Κεφάλαιο 1 - Εισαγωγή

Ένα P2P (peer-to-peer) δίκτυο είναι ένα δίκτυο διασυνδεδεμένων υπολογιστών που δεν ακολουθεί την συνηθισμένη αρχιτεκτονική πελάτη-εξυπηρετητή, όπου έχουμε κάποιους εξειδικευμένους υπολογιστές που εξυπηρετούν τις αιτήσεις των υπολοίπων. Πρόκειται για ομότιμα δίκτυα στα οποία κάθε χρήστης λειτουργεί ταυτόχρονα ως πελάτης και ως εξυπηρετητής.

Το μοντέλο αυτό, λοιπόν, είναι διαφορετικό από αυτό που έχουμε συνηθίσει τα τελευταία χρόνια και το οποίο βασίζεται στην ύπαρξη ενός σταθερού εξυπηρετητή και αποτελεί τον κορμό του Internet όπως το γνωρίζουμε σήμερα. Σε γενικές γραμμές, με χαρακτηριστικό παράδειγμα το http, οι χρήστες του Internet ζητούν κάποιο αντικείμενο από κάποιον εξυπηρετητή, ο οποίος λειτουργεί ειδικά για το σκοπό αυτό. Σε ένα P2P δίκτυο, όλοι οι χρήστες συμμετέχουν ενεργά στο σύστημα παρέχοντας πόρους, όπως bandwidth, αποθηκευτικό χώρο και επεξεργαστική ισχύ. Το γεγονός αυτό έχει ως αποτέλεσμα η αύξηση του μεγέθους του δικτύου να αυξάνει και την διαθεσιμότητα πόρων.

Τα τελευταία χρόνια η διάδοση των P2P δικτύων είναι ταχύτατη, με πιο σημαντική και μαζική την χρήση τους για τη διακίνηση αρχείων. Ενδεικτικό της αποδοχής των εφαρμογών αυτών είναι το γεγονός ότι, όπως φαίνεται στο παρακάτω γράφημα, στις μέρες μας η P2P κίνηση καταλαμβάνει περίπου το 60% της συνολικής κίνησης στο Internet. Επίσης, είναι πασιφανής ο ταχύτατος ρυθμός διάδοσης των εφαρμογών αυτών, ενώ βλέπουμε ότι έχουν υποσκελίσει παραδοσιακές χρήσεις του Internet όπως το world wide web.



Εικόνα 1: Κατανομή διαδικτύακης κίνησης

Παρότι όμως η διακίνηση αρχείων είναι η πιο διαδεδομένη εφαρμογή των P2P δικτύων δεν είναι και η μοναδική. Στη συνέχεια περιγράφουμε αναλυτικότερα κάποιες από τις εφαρμογές τους.

- **Content Distribution**

Όπως αναφέρθηκε και προηγουμένως πρόκειται για την πιο διαδεδομένη εφαρμογή των P2P δικτύων. Τα προγράμματα αυτά δίνουν την δυνατότητα στους χρήστες να ανταλλάσσουν δεδομένα με μεγάλη ταχύτητα και παρέχουν αυξημένη διαθεσιμότητα. Στην κατηγορία αυτή περιλαμβάνονται οι γνωστές εφαρμογές διαμοιρασμού αρχείων καθώς και πιο εξεζητημένες εφαρμογές που επιτρέπουν την κατανεμημένη αποθήκευση σημαντικών αρχείων με αποτέλεσμα την καλύτερη και ασφαλέστερη διαχείριση τους. Χαρακτηριστικός εκπρόσωπος αυτής της κατηγορίας αυτής είναι το BitTorrent.

- **Communication and Collaboration**

Στην κατηγορία αυτή περιέχονται τα προγράμματα chat και instant messaging, που επιτρέπουν, συνήθως σε πραγματικό χρόνο, την συνεργασία και την επικοινωνία ανάμεσα στους χρήστες. Χαρακτηριστικό παράδειγμα είναι το Skype.

- **Distributed Computing**

Στην κατηγορία αυτή περιέχονται προγράμματα που στοχεύουν στην αξιοποίηση της υπολογιστικής ισχύος των peers. Χρησιμοποιούνται για την επίλυση πολύπλοκων προβλημάτων τα οποία είναι διαχωρίσιμα σε απλούστερα, τα οποία επιλύονται από κάθε peer, ο οποίος στη συνέχεια επιστρέφει τα αποτελέσματα. Σε αυτές τις εφαρμογές απαιτείται η ύπαρξη κεντρικού ελέγχου για τον διαμοιρασμό των εργασιών και τη συλλογή των αποτελεσμάτων. Χαρακτηριστικό παράδειγμα είναι το Folding@home που ασχολείται με την προσομοίωση της αναδίπλωσης των πρωτεϊνών.

- **P2P Streaming**

Στην κατηγορία αυτή περιέχονται προγράμματα που χρησιμοποιούν P2P πρωτόκολλα για το streaming δεδομένων ήχου και εικόνας. Πρόκειται στην ουσία για την δημιουργία τηλεοπτικών και ραδιοφωνικών σταθμών που εκμεταλλεύονται την P2P αρχιτεκτονική για την εκπομπή του προγράμματος τους. Ενδεικτικό παράδειγμα τέτοιου προγράμματος είναι το PPLive.

Κεφάλαιο 2 – Βιβλιογραφική Έρευνα

Το γεγονός ότι τα Peer-to-Peer δίκτυα χρησιμοποιούνται σήμερα σε πολλές διαφορετικές εφαρμογές καθιστά δύσκολη τη μοντελοποίησή τους. Επιπλέον, η έμφυτη πολυπλοκότητα που παρουσιάζουν λόγω του μεγέθους τους αποτελεί μια ακόμη πρόκληση. Μέχρι σήμερα έχουν γίνει αρκετές αξιόλογες προσπάθειες μοντελοποίησης των Peer-to-Peer δικτύων που αξίζει να αναφέρουμε.

Στο [1] οι συγγραφείς εξετάζουν το πρόβλημα της βέλτιστης επιλογής κόμβων σε δύο πολύ σημαντικές περιπτώσεις: το κατέβασμα ενός αρχείου και τη αναπαραγωγή ενός αρχείου βίντεο ή ήχου που βρίσκεται αποθηκευμένο σε κάποιο άλλο υπολογιστή (streaming). Οι συγγραφείς οραματίζονται ένα Peer-to-Peer δίκτυο που θα λειτουργεί σαν μια αγορά πόρων όπου κάθε χρήστης πουλά τους πόρους που διαθέτει και αγοράζει πόρους από άλλους χρήστες με βάση κάποιες συναρτήσεις κόστους. Υπό αυτές τις συνθήκες, το πρόβλημα της βέλτιστης επιλογής κόμβων αναφέρεται στην επιλογή εκείνων των κόμβων που διαθέτουν το ζητούμενο αρχείο αλλά και εκείνου του ρυθμού κατεβάσματος από κάθε κόμβο ώστε να ελαχιστοποιείται το κόστος απόκτησής του. Ειδικά, στην περίπτωση του streaming το πρόβλημα λύνεται χρησιμοποιώντας τον περιορισμό ότι θα πρέπει να εξασφαλίζεται συνεχής αναπαραγωγή του αρχείου ακόμη και αν αποτύχουν οι συνδέσεις με κάποιο αριθμό κόμβων.

Στο [2] εξετάζεται με βάση απλά μοντέλα το δημοφιλές πρωτόκολλο ανταλλαγής αρχείων BitTorrent ως προς την αποδοτικότητά του. Ελέγχεται η αποτελεσματικότητα του ενσωματωμένου μηχανισμού προσφοράς κινήτρων στους χρήστες ώστε να συνεισφέρουν στο δίκτυο, η αποτελεσματικότητα με την οποία διανέμονται τα αρχεία στους χρήστες αλλά και ο τρόπος που συμπεριφέρεται το πρωτόκολλο καθώς αυξάνεται πολύ ο αριθμός των κόμβων στο δίκτυο. Όσον αφορά την αποτελεσματικότητα στον διαμοιρασμό των αρχείων, οι συγγραφείς προσδιορίζουν την πιθανότητα ένας κόμβος να βρει τα κομμάτια ενός αρχείου που χρειάζεται με βάση κάποιο μέγιστο αριθμό συνδέσεων και αποδεικνύουν ότι αυτή είναι ανεξάρτητη από το ρυθμό άφιξης νέων αιτήσεων και, στις περισσότερες περιπτώσεις, πολύ κοντά στη μονάδα.

Οι συγγραφείς του [3] προτείνουν έναν απλό πρωτόκολλο διανομής αρχείων όπου κάθε αρχείο χωρίζεται σε κομμάτια και κάθε χρήστης που κατεβάζει κάθε κομμάτι μπορεί να το δίνει σε άλλους χωρίς να είναι απαραίτητο να έχει πάρει όλο το αρχείο. Το πρωτόκολλο αποτελείται από ένα συνδυασμό push και pull μηχανισμών, που εφαρμόζονται από κάθε κόμβο σε διαφορετικό στάδιο της διανομής του αρχείου ώστε να επιτευχθεί καλύτερη απόδοση. Επίσης, τονίζουν τη μεγάλη σημασία της σωστής επιλογής του κομματιού που θα ζητήσει ένας κόμβος κάθε στιγμή στην ελαχιστοποίηση του χρόνου διανομής του αρχείου. Τέλος, προσδιορίζεται ο χρόνος που απαιτείται από το πρωτόκολλο για τη διανομή ενός μεγάλου μέρους του αρχείου σε όλους τους χρήστες και αποδεικνύεται πόσο βοηθά τη διαδικασία διανομής το σπάσιμο του αρχείου σε κομμάτια.

Στο [4] εξετάζεται το πρόβλημα της δρομολόγησης διαφορετικών κατηγοριών από πελάτες σε πολλούς κατανεμημένους εξυπηρετητές ώστε να ελαχιστοποιηθεί ο μέσος χρόνος ανάκτησης κάθε κατηγορίας πελατών. Το πρόβλημα αυτό βρίσκει εφαρμογή σε πολλά σύγχρονα κατανεμημένα συστήματα. Στο πρόβλημα της εύρεσης των βέλτιστων προτεραιοτήτων στο δίκτυο αποδεικνύεται ότι η βέλτιστη επιλογή είναι αυτή που επιβάλλεται από τον μC κανόνα. Επίσης, το παραπάνω πρόβλημα ορίζεται ως ένα μη-γραμμικό πρόβλημα βελτιστοποίησης και αποδεικνύοντας ότι κάθε

εσωτερικό τοπικό ελάχιστο αποτελεί και ολικό ελάχιστο διευκολύνεται η διαδικασία επίλυσής του. Τέλος, με βάση το μαθηματικό μοντέλο προτείνονται κάποιοι απλοί προσεγγιστικοί αλγόριθμοι με απόδοση πολύ κοντά στα θεωρητικά αποτελέσματα.

Στο [5] ο συγγραφέας αναλύει βασικές έννοιες των ουρών δίνοντας ιδιαίτερη έμφαση σε θέματα απόδοσης των συστημάτων που καθορίζονται από παράγοντες όπως οι φυσικές παράμετροι του συστήματος (ταχύτητα εξυπηρέτησης, αριθμός εξυπηρετητών κ.α.), τα χαρακτηριστικά της κίνησης (η κατανομή των αφίξεων, τα μεγέθη των αιτημάτων κ.α.) και ο τρόπος εξυπηρέτησης των αιτήσεων. Ειδικά για την περίπτωση της M/G/1 ουράς, αποδεικνύονται οι τύποι που δίνουν τους χρόνους αναμονής και ανάκτησης ενώ εξετάζεται και η περίπτωση των ουρών πολλαπλών κλάσεων με προτεραιότητες. Ακόμη, καλύπτονται οι περιπτώσεις των preemptive αλλά και non-preemptive μεθόδων παραχώρησης προτεραιότητας καθώς και η περίπτωση της γενικής ουράς G/G/1.

Στο [6] γίνεται μια προσπάθεια κατανόησης της Peer-to-Peer εφαρμογής ανταλλαγής αρχείων KaZaA. Το KaZaA είναι σήμερα μια από τις σημαντικότερες εφαρμογές του internet με περισσότερους από 3 εκατομμύρια χρήστες. Όμως το γεγονός ότι πρόκειται για ένα εμπορικό πρόγραμμα που χρησιμοποιεί κρυπτογράφηση είναι πολύ δύσκολο να μάθουμε τον τρόπο λειτουργίας του. Οι συγγραφείς σε μια προσπάθεια να αποκαλυφθούν τα βασικά χαρακτηριστικά του KaZaA χρησιμοποίησαν κάποια εργαλεία παρακολούθησης δικτύου ώστε να μελετήσουν τα μηνύματα που ανταλλάσσονται μεταξύ των κόμβων από το KaZaA. Με τον τρόπο αυτό κατάφεραν να βγάλουν σημαντικά συμπεράσματα για τη δομή και την λειτουργία του.

Οι συγγραφείς του [7] πραγματεύονται το πρόβλημα της διανομής ενός αρχείου σε ένα δίκτυο όπου υπάρχουν κόμβοι που είτε το έχουν ολόκληρο (seeders) είτε έχουν ένα μέρος του και θέλουν να κατεβάσουν και το υπόλοιπο (leechers), οι οποίοι όμως συνεισφέρουν και αυτοί στο διαμοιρασμό του. Σκοπός είναι η διανομή ολόκληρου το αρχείου στους leechers στο λιγότερο δυνατό χρόνο. Στη δημοσίευση αυτή παρουσιάζονται τύποι για τον ελάχιστο χρόνο διανομής του αρχείου σε ένα τέτοιο δίκτυο, οι οποίοι επαληθεύονται και πρακτικά με παρατήρηση πραγματικών δικτύων. Οι τύποι αυτοί εξαρτώνται από το μέγεθος του αρχείου, τους ρυθμούς αποστολής δεδομένων των seeders και τους ρυθμούς με τους οποίους μπορούν να ανεβάζουν και να κατεβάζουν δεδομένα οι leechers, Επίσης, εξετάζεται η περίπτωση όπου οι κόμβοι που θέλουν να κατεβάσουν το αρχείο χωρίζονται σε δύο κατηγορίες με διαφορετική προτεραιότητα. Συγκεκριμένα, εξετάζεται ο τρόπος με τον οποίο οι κόμβοι με την μικρότερη προτεραιότητα μπορούν να συνεισφέρουν σημαντικά στην ταχύτερη διανομή του αρχείου στους κόμβους με μεγαλύτερη προτεραιότητα.

Ο συγγραφέας του [15] εξετάζει το πρόβλημα ελαχιστοποίησης της μέσης και της μέγιστης καθυστέρησης ανάκτησης ενός αρχείου χρησιμοποιώντας κατανεμημένες υλοποιήσεις που βασίζονται στην εγωιστική συμπεριφορά των κόμβων. Στην παρούσα εργασία συγκρίνουμε την προσέγγιση αυτή με την αλτρουιστική προσέγγιση που προτείνουμε.

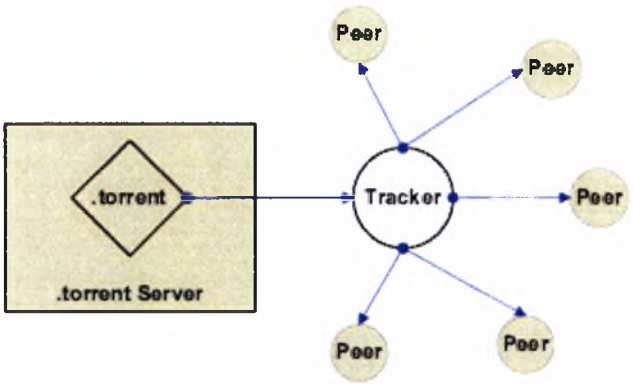
Στην συνέχεια θα παρουσιάσουμε αναλυτικά το πρωτόκολλο BitTorrent καθώς οι προτεινόμενοι αλγόριθμοι μπορούν να υλοποιηθούν και να ενσωματωθούν εύκολα στο πρωτόκολλο αυτό.

Το BitTorrent αποτελεί ένα Peer – to – Peer πρωτόκολλο δεύτερης γενιάς που αναπτύχθηκε το 2002 από τον Bram Cohen για να παρέχει γρήγορο διαμοιρασμό κυρίως δημοφιλών αρχείων. Τυπικά, για ένα δημοφιλές αρχείο υπάρχουν μερικές χιλιάδες κόμβων που το κατεβάζουν ταυτόχρονα ενώ κατά τη διάρκεια ζωής του ο συνολικός αριθμός χρηστών που θα το κατεβάσουν μπορεί να είναι δεκάδες ή ακόμη και εκατοντάδες χιλιάδες.

Η βασική ιδέα στην οποία στηρίζεται το BitTorrent είναι ο χωρισμός ενός αρχείου σε κομμάτια μεγέθους 256KB το καθένα. Όταν κάποιος κόμβος θέλει, λοιπόν, να κατεβάσει κάποιο αρχείο πρέπει να συνδεθεί ταυτόχρονα με πολλούς κόμβους που έχουν τα κομμάτια του και να τα κατεβάσει. Σύμφωνα με το πρωτόκολλο, υπάρχουν δύο κατηγορίες κόμβων, αυτοί που έχουν όλα τα κομμάτια που αποτελούν το αρχείο, οι οποίοι λέγονται seeders, και εκείνοι που έχουν ορισμένα από τα κομμάτια που το αποτελούν, και λέγονται downloaders. Σκοπός των seeders είναι να παραμένουν στο δίκτυο ώστε να δίνουν τα κομμάτια που αποτελούν το αρχείο σε άλλους, επομένως οι κόμβοι αυτοί μόνο ανεβάζουν δεδομένα. Οι downloaders, αντίθετα, κατεβάζουν αλλά και ανεβάζουν δεδομένα. Όπως ορίζει το πρωτόκολλο, κάθε κόμβος μπορεί να συνεισφέρει στον διαμοιρασμό ενός κομματιού ενός αρχείου μόλις ολοκληρωθεί το κατέβασμα του κομματιού αυτού. Αυτό σημαίνει ότι ένας downloader μπορεί να ανεβάζει όλα τα τμήματα τα οποία έχει κατεβάσει βοηθώντας έτσι το έργο των seeders. Το πρωτόκολλο προσπαθεί, εφαρμόζοντας την πολιτική «Το σπανιότερο κομμάτι πρώτο» να εξασφαλίσει ομοιόμορφή κατανομή των κομματιών στο δίκτυο έτσι ώστε να εξασφαλίζεται ομαλός διαμοιρασμός όλων των κομματιών ενός αρχείου.

Για την διευκόλυνση αυτής της διαδικασίας, το BitTorrent χρησιμοποιεί ένα μηχανισμό που ονομάζεται tracker και που φαίνεται σχηματικά στην Εικόνα 2. Όταν ένας κόμβος θέλει να κατεβάσει ένα αρχείο πρέπει με κάποιο τρόπο να συνδεθεί στον tracker του αρχείου αυτού. Για να το κάνει αυτό, κατεβάζει ένα αρχείο με κατάληξη .torrent. Το αρχείο αυτό περιέχει πληροφορίες σχετικά με το αρχείο, πρόκειται δηλαδή για ένα meta-data αρχείο. Οι πληροφορίες που περιέχει είναι το μέγεθος του αρχείου, το όνομά του, κάποιες πληροφορίες κατακερματισμού, και τη διεύθυνση του tracker. Ο tracker διατηρεί καταγεγραμμένους όλους τους κόμβους που έχουν το αρχείο αυτό, είτε ολόκληρο είτε ένα τμήμα του και χρησιμοποιεί ένα απλό πρωτόκολλο που βασίζεται στο HTTP. Με βάση αυτό κάθε κόμβος στέλνει ένα μήνυμα με πληροφορίες για το αρχείο που κατεβάζει και έναν αριθμό port στην οποία θα γίνουν οι συνδέσεις. Στη συνέχεια, ο tracker απαντά με μια τυχαία λίστα των κόμβων που έχουν κατεβάσει ή κατεβάζουν εκείνη τη στιγμή το αρχείο. Ο κόμβος τότε συνδέεται με αυτούς και μαθαίνει ποια κομμάτια του αρχείου έχει κάθε ένας. Έπειτα, ο κόμβος ζητά από τους άλλους όλα τα κομμάτια τα οποία δεν έχει. Όταν τελειώσει το κατέβασμα ενός κομματιού εφαρμόζει σε αυτό μια συνάρτηση κατακερματισμού για να ελέγξει αν το κατέβασε σωστά και στη συνέχεια ανακοινώνει σε όλους τους κόμβους με τους οποίους είναι συνδεδεμένος ότι έχει πάρει το κομμάτι αυτό και μπορεί στο εξής να το ανεβάζει και σε άλλους. Κάθε κόμβος μπορεί να στέλνει δεδομένα ταυτόχρονα σε περιορισμένο αριθμό χρηστών, που συνήθως ορίζεται στους τέσσερις. Οι κόμβοι που θα επιλεγούν για ανέβασμα καθορίζονται από τον ρυθμό κατεβάσματος από αυτούς εκείνη τη στιγμή. Κάθε κόμβος ανεβάζει στους τέσσερις κόμβους οι οποίοι του δίνουν δεδομένα με μεγαλύτερο ρυθμό. Οι αιτήσεις για ανέβασμα που φτάνουν από άλλους κόμβους μπαίνουν σε μία ουρά και εξυπηρετούνται αργότερα. Με τον τρόπο αυτό ενθαρρύνονται οι κόμβοι να ανεβάζουν δεδομένα, αντιμετωπίζοντας έτσι το φαινόμενο free-riding, όπου ένας

κόμβος ενδιαφέρεται μόνο να κατεβάσει ένα αρχείο χωρίς να βοηθά και ο ίδιος στο διαμοιρασμό του.



Εικόνα 2: Η αρχιτεκτονική BitTorrent αποτελούμενη από τον tracker, το αρχείο .torrent και τους κόμβους

Το φαινόμενο κατά το οποίο ένας κόμβος αρνείται προσωρινά να ανεβάσει δεδομένα σε κάποιους άλλους λέγεται choking. Αυτό μπορεί να συμβεί για πολλούς λόγους, όπως το γεγονός ότι το πρωτόκολλο TCP δεν αποδίδει καλά όταν υπάρχουν πολλές συνδέσεις, οπότε πρέπει να υπάρχει ένας μέγιστος αριθμός συνδέσεων. Είναι πολύ σημαντικό για κάποιο κόμβο να εφαρμόζει ένα καλό αλγόριθμο choking έτσι ώστε να αποφεύγονται συνεχή ανοίγματα και κλεισίματα συνδέσεων, καθώς καταναλώνουν μέρος των πόρων, αλλά και να εξασφαλίζεται σύνδεση σε κόμβους που θα του επιτρέψουν να κατεβάσει με το μεγαλύτερο δυνατό ρυθμό. Για την επίτευξη του τελευταίου κάθε κόμβος εφαρμόζει έναν αλγόριθμο, που ονομάζεται optimistic unchoking. Αφού κάθε κόμβος, έστω ο Α, ανεβάζει μόνο σε τέσσερις κόμβους, εκείνους που το δίνουν εκείνη τη στιγμή με τη μεγαλύτερη ταχύτητα, είναι δυνατό να υπάρχει κάποιος άλλος κόμβος Β που αν ανέβαζε σε αυτόν ο Α, να μπορούσε να του εξασφαλίσει μεγαλύτερο ρυθμό κατεβάσματος από κάθε άλλο κόμβο στον οποίο ανεβάζει αυτή τη στιγμή ο Α. Για να μπορεί, λοιπόν, κάθε κόμβος να ανακαλύπτει το ρυθμό κατεβάσματος από άλλους κόμβους επιλέγει τυχαία έναν πέμπτο κόμβο, που του έχει ζητήσει κάποιο κομμάτι που διαθέτει, και τον εξυπηρετεί. Αφού αντιληφθεί το ρυθμό με τον οποίο μπορεί αυτός ο κόμβος να του δίνει δεδομένα, διακόπτει το ανέβασμα σε εκείνο από τους πέντε με το μικρότερο ρυθμό. Αυτή η διαδικασία επαναλαμβάνεται κάθε 30 δευτερόλεπτα και έτσι επιτυγχάνεται ο μέγιστος ρυθμός κατεβάσματος.

Η αναζήτηση των αρχείων καθώς και η διανομή των .torrent αρχείων γίνεται μέσω δικτυακών τόπων (SuprNova.org, PirateBay.org, TorrentSpy.com κ.α). Κάθε χρήστης μπορεί να αναζητήσει το αρχείο που θέλει, να κατεβάσει το .torrent αρχείο που αναφέρεται σε αυτό και να το ανοίξει με ένα από τα πολλά διαθέσιμα συμβατά με το πρωτόκολλο προγράμματα (μTorrent, BitComet, ShareAza κ.α.). Στη συνέχεια το πρόγραμμα θα αναλάβει να συνδεθεί με τον tracker και να κατεβάσει όλα τα κομμάτια από τα οποία αποτελείται το αρχείο.

Σύμφωνα με μετρήσεις της κίνησης στις κεντρικές αρτηρίες του Internet τον Ιούνιου 2004, το ποσοστό της κίνησης που χρησιμοποιεί το πρωτόκολλο BitTorrent ανερχόταν στο 53% της συνολικής P2P κίνησης. Η τεράστια αυτή διάδοση του BitTorrent οφείλεται σε κάποια πολύ σημαντικά πλεονεκτήματα που έχει. Το βασικό πλεονέκτημά του είναι το γεγονός ότι στο διαμοιρασμό των αρχείων μπορούν να συνεισφέρουν ακόμη και οι κόμβοι που δεν έχουν ολοκληρώσει το κατέβασμά του. Με τον τρόπο αυτό μειώνονται οι απαιτήσεις σε bandwidth από τους seeders, αφού

υποβοηθούνται και από τους lechers, αλλά και γίνεται και γρηγορότερος ο διαμοιρασμός του αρχείου, αφού όταν ο seeder δίνει ένα κομμάτι σε κάποιο downloader αυξάνει ταυτόχρονα και την διαθεσιμότητά του. Επιπλέον, η χρήση συναρτήσεων κατακερματισμού για τον έλεγχο των κομματιών βοηθά στο σωστότερο διαμοιρασμό των αρχείων.

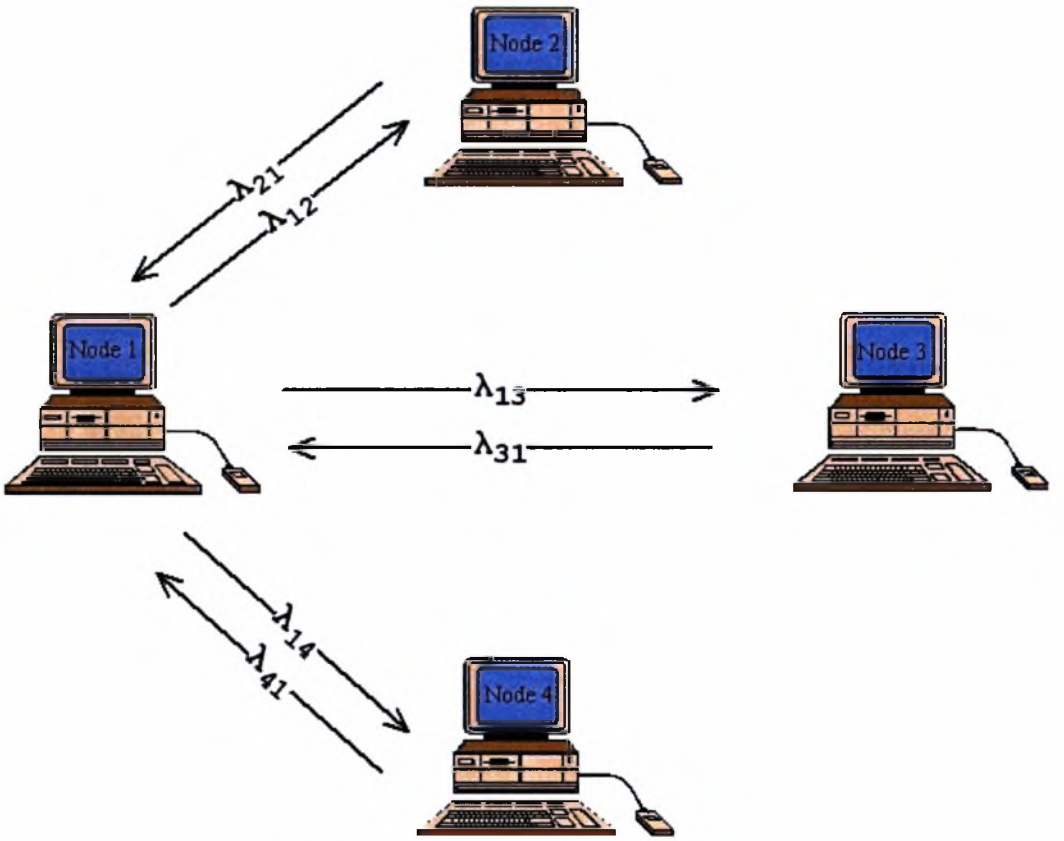
Κεφάλαιο 3 – Το Μοντέλο του συστήματος

Θεωρούμε ένα peer-to-peer σύστημα που αποτελείται από N κόμβους, καθένας εκ των οποίων έχει ένα σύνολο αρχείων προς διάθεση. Τα αρχεία αυτά μπορεί να είναι οποιασδήποτε μορφής και το μέγεθος τους γενικά κυμαίνεται από μερικά KB μέχρι και μερικά GB. Η σύσταση του συστήματος είναι σταθερή, θεωρούμε δηλαδή ότι δεν εισέρχονται ούτε αποχωρούν κόμβοι από το σύστημα. Κάθε κόμβος προσφέρει ένα αρχείο του απαντώντας σε αιτήσεις των άλλων χρηστών ενώ ταυτόχρονα ζητά αρχεία από τους άλλους. Κάθε κόμβος λοιπόν, λειτουργεί ταυτόχρονα και ως εξυπηρετητής και ως πελάτης. Αυτό το σύνολο κόμβων θα μπορούσε να προκύπτει από ένα μηχανισμό εύρεσης αρχείου ενός peer-to-peer προγράμματος, όπως ο μηχανισμός του BitTorrent .

Συμβολίζουμε τον συνολικό ρυθμό παραγωγής αιτήσεων του κόμβου i με λ_i που δηλώνει τον αριθμό των αιτήσεων για διαφορετικά αντικείμενα ανά μονάδα χρόνου. Θεωρούμε ότι η παραγωγή αιτήσεων κάθε κόμβου i ακολουθεί κατανομή Poisson, ενώ το μέγεθος των αιτήσεων είναι εκθετική τυχαία μεταβλητή με μέση τιμή s_i. Επομένως ο συνολικός ρυθμός παραγωγής αιτήσεων κάθε κόμβου i σε bits/sec είναι $\lambda_i s_i$. Οι N κόμβοι σχηματίζουν ένα κλειστό δίκτυο, όπου κάθε αίτηση ενός κόμβου μπορεί να εξυπηρετηθεί από ένα υποσύνολο των άλλων κόμβων. Συμβολίζουμε, λοιπόν, με λ_{ij} το ρυθμό παραγωγής αιτήσεων του i που ανατίθεται στον j. Ο διαμοιρασμός του ρυθμού παραγωγής αιτήσεων του πελάτη i μπορεί να αναπαρασταθεί με ένα διάνυσμα $\boldsymbol{\lambda}_i = \left(\lambda_{ij} : j = 1,\ldots N, j \neq i\right)$ με $\lambda_i = \sum_{j=1, j\neq i}^{N} \lambda_{ij}$. Εναλλακτικά μπορεί να γραφεί ως ένα διάνυσμα πιθανοτήτων $\mathbf{p}_i = \left(p_{ij} : j = 1,\ldots N, j \neq i\right)$ όπου $p_{ij} = \frac{\lambda_{ij}}{\lambda_i}$ είναι το ποσοστό των αιτήσεων του κόμβου i που στέλνονται στον j. Ο διαμοιρασμός αυτός των αιτήσεων μπορεί επίσης να αναπαριστά την περίπτωση που ένας χρήστης ανακτά τμήματα του ίδιου αντικειμένου από διάφορους κόμβους. Ο διαμοιρασμός των ρυθμών παραγωγής των αιτήσεων όλου του συστήματος μπορεί να αναπαρασταθεί με έναν πίνακα $\boldsymbol{\Lambda}$ που η i-οστή γραμμή του είναι το διάνυσμα $\boldsymbol{\lambda}_i$ και τα στοιχεία της διαγωνίου είναι μηδενικά. Εναλλακτικά, μπορεί να αναπαρασταθεί με ένα πίνακα πιθανοτήτων $\boldsymbol{P}$ που η i-οστή γραμμή του είναι το διάνυσμα $\mathbf{p}_i$.

Κάθε κόμβος j ως εξυπηρετητής δέχεται αιτήσεις, τις οποίες πρέπει να εξυπηρετήσει, με ρυθμό λ_{ij} από τους άλλους κόμβους i≠j. Κάθε εξυπηρετητής j χαρακτηρίζεται από τον ρυθμό εξυπηρέτησης C_j. Επομένως, ο μέσος χρόνος εξυπηρέτησης του i από τον j είναι s_i / C_j.

Το μοντέλο αυτό απεικονίζεται στην παρακάτω εικόνα για ένα δίκτυο τεσσάρων κόμβων.



Εικόνα 3: Το δίκτυο για τέσσερις κόμβους

Ευστάθεια: Η αναγκαία και ικανή συνθήκη για ευστάθεια (δηλαδή, για πεπερασμένη καθυστέρηση) είναι $\sum_{i=1, i\neq j}^{N} \lambda_{ij} s_i < C_j$ για κάθε εξυπηρετητή $j=1,...N$. Το σύστημα είναι ευσταθές εάν $\sum_{i=1}^{N} \lambda_i s_i < \sum_{j=1}^{N} C_j$,συνθήκες οι οποίες θεωρούμε ότι ισχύουν.

Αν υποθέσουμε ότι το μέγεθος αρχείων s που αιτούνται οι πελάτες ακολουθεί εκθετική κατανομή με την ίδια μέση τιμή για όλους, τότε ο μέσος ρυθμός εξυπηρέτησης στον εξυπηρετητή j είναι $\mu_j = s_i / C_j$ και οι χρόνοι εξυπηρέτησης ακολουθούν εκθετική κατανομή. Βάσει αυτής της υπόθεσης, κάθε εξυπηρετητής μπορεί να μοντελοποιηθεί ως μια M/M/1 ουρά με ρυθμό εξυπηρέτησης μ_j. Οι εξυπηρετητές εφαρμόζουν μια preemptive πολιτική εξυπηρέτησης, όπου η εξυπηρέτηση γίνεται βάσει προτεραιοτήτων και με το σύνολο της χωρητικότητας του εξυπηρετητή. Η εξυπηρέτηση μιας αίτησης διακόπτεται αν υπάρξει άφιξη αίτησης με μεγαλύτερη προτεραιότητα και συνεχίζεται από το σημείο που διακόπηκε μόλις εξυπηρετηθούν όλοι ο χρήστες μεγαλύτερης προτεραιότητας. Η ανάθεση προτεραιοτήτων από έναν εξυπηρετητή j προσδιορίζεται από ένα διάνυσμα στήλη $\boldsymbol{\pi}_j$. Το j-οστό στοιχείο του διανύσματος είναι μηδέν ενώ τα υπόλοιπα στοιχεία $\pi_j(i)$

αποτελούν μία αναδιάταξη του *{1,....,N}|{j}* τέτοια ώστε αν $\pi_j(i) < \pi_j(k)$, τότε ο χρήστης i απολαμβάνει μεγαλύτερης προτεραιότητας του k στον εξυπηρετητή j. Με ***Π*** συμβολίζουμε τον πίνακα προτεραιοτήτων του δικτύου, που έχει στη στήλη j το διάνυσμα $\boldsymbol{\pi}_j$. Ο μέσος χρόνος ανάκτησης των αιτήσεων του i με προτεραιότητα $\pi_j(i)$ στον εξυπηρετητή j δίνεται από τον τύπο

$$D_{ij} = \frac{1}{\mu_j \left(1 - \sum_{k:\pi_j(k)>\pi_j(i)} \rho_{kj}\right)\left(1 - \sum_{k:\pi_j(k)>\pi_j(i)} \rho_{kj}\right)} \qquad (3.1)$$

όπου $\rho_{ij} = \frac{\lambda_{ij}}{\mu_j}$ είναι ο φόρτος του i στον j. Ο παραπάνω χρόνος ανάκτησης περιλαμβάνει τον χρόνο αναμονής στην ουρά και τον χρόνο εξυπηρέτησης.

Είναι εμφανές ότι η αναμονή D_{ij} εξαρτάται από την προτεραιότητα που απολαμβάνει ο i όταν εξυπηρετείται από τον j, καθώς και από τα σπασίματα λ_{ij}. Η μέση καθυστέρηση του χρήστη i στο σύνολο των άλλων κόμβων είναι $D_i = \sum_{j=1, j\neq i}^{N} p_{ij} D_{ij}$, ενώ η μέγιστη καθυστέρηση είναι $D_i = \max_{\lambda_{ij}} p_{ij} D_{ij}$. Επομένως, κάθε χρήστης μπορεί να επηρεάσει το μέσο χρόνο αναμονής του μόνο μερικώς, ελέγχοντας τα σπασίματα του.

Κεφάλαιο 4 – Ελαχιστοποίηση μέσου χρόνου ανάκτησης

Θεωρούμε ότι σε κάθε κόμβο i αντιστοιχίζεται ένα θετικό βάρος b_i. Αρχικά πραγματευόμαστε την εύρεση ενός εφικτού πίνακα σπασιμάτων $\boldsymbol{\Lambda}$ και ενός πίνακα προτεραιοτήτων $\boldsymbol{\Pi}$ που λύνει το παρακάτω πρόβλημα

$$\min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} D_i = \min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \sum_{j=1, j\neq i}^{N} \frac{\lambda_{ij}}{\lambda_i} D_{ij}(\Lambda,\Pi) = \min_{\Lambda,\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) \qquad (4.1)$$

Κάθε εξυπηρετητής επηρεάζει τους χρόνους αναμονής των κόμβων ανάλογα με την προτεραιότητα που δίνει στον καθένα. Κάθε πελάτης i επηρεάζει μερικώς την δικιά του καθυστέρηση επιλέγοντας τα σπασίματα των αιτήσεων του στους εξυπηρετητές. Ωστόσο, με αυτό τον τρόπο επηρεάζει και τις καθυστερήσεις των κόμβων χαμηλότερης προτεραιότητας όπως φαίνεται από την έκφραση του D_{ij}. Είναι εμφανές ότι το παραπάνω πρόβλημα είναι η εύρεση της κοινωνικά βέλτιστης λύσης.

Το παραπάνω πρόβλημα ελαχιστοποίησης μπορεί ισοδύναμα να γραφεί ως:

$$\min_{\Lambda,\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) = \min_{\Lambda} \min_{\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) = \min_{\Lambda} \sum_{j=1}^{N} \min_{\pi(j)} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) \qquad (4.2)$$

Με άλλα λόγια, για ένα δεδομένο πίνακα σπασιμάτων, το πρόβλημα της εύρεσης του πίνακα προτεραιοτήτων μπορεί να αναλυθεί σε ανεξάρτητα προβλήματα για κάθε εξυπερετητή. Το καθένα εκ των προβλημάτων είναι η εύρεση του διανύσματος προτεραιοτήτων που ελαχιστοποιεί το βεβαρημένο άθροισμα των χρόνων αναμονής στον συγκεκριμένο εξυπηρετητή.

Η λύση αυτού του προβλήματος δίνεται από τον μc κανόνα. Σύμφωνα με αυτόν, η πολιτική που ελαχιστοποιεί την συνολική βεβαρημένη καθυστέρηση είναι να δίνεις προτεραιότητα κατά φθίνουσα σειρά βάσει του γινομένου βάρους και ρυθμού εξυπηρέτησης, Η απόδειξη βασίζεται στην παρακάτω ιδέα: Ξεκινώντας από μια υποθετικά βέλτιστη λύση εναλλάσσουμε τις προτεραιότητες δύο διαδοχικών, ως προς την προτεραιότητα, πελατών και χρησιμοποιώντας τον Conservation law εξάγουμε την παραπάνω συνθήκη για την βέλτιστη λύση (βλ. [5]). Στην περίπτωσή μας τα βάρη είναι τα b_i και οι ρυθμοί εξυπηρέτησης μ_j.

Βάσει του παραπάνω αποτελέσματος η βέλτιστη ανάθεση προτεραιοτήτων είναι γνωστή, και μάλιστα είναι τέτοια ώστε κάθε χρήστης να απολαμβάνει την ίδια προτεραιότητα σε κάθε εξυπηρετητή, πρόκειται δηλαδή για μια καθολική διάταξη για όλο το σύστημα. Επομένως, το πρόβλημα ελαχιστοποίησης περιορίζεται στην εύρεση του βέλτιστου εφικτού πίνακα σπασιμάτων $\boldsymbol{\Lambda}$.

Το πρόβλημα αυτό μπορεί εύκολα να λυθεί με κεντρικοποιημένο τρόπο. Στη συνέχεια θα παρουσιάσουμε δύο αλγορίθμους για την επίλυση του παραπάνω προβλήματος ελαχιστοποίησης με κατανεμημένο τρόπο. Κάθε πελάτης i καθορίζει το δικό του διάνυσμα σπασιμάτων $\boldsymbol{\lambda_i}$ βάσει ενός κριτηρίου βελτιστοποίησης. Οι αλγόριθμοι βασίζονται στην αλτρουιστική προσέγγιση καθώς κάθε χρήστης συνυπολογίζει και την επίδραση που έχει στις καθυστερήσεις των άλλων χρηστών. Για την επίλυση του προβλήματος ελαχιστοποίησης, μπορεί να χρησιμοποιηθεί η μέθοδος Waterfilling, την οποία αναλύουμε στο επόμενο εδάφιο.

Η μέθοδος Waterfilling

Η μέθοδος αυτή χρησιμοποιείται για την επίλυση προβλημάτων της μορφής:

$$\min_{\lambda_{ij}} F\left(\lambda_{ij}\right) \qquad (4.3)$$
$$s.t. \qquad \sum_j \lambda_{ij} = \lambda_i$$

Δηλαδή πρόκειται για προβλήματα στα οποία θέλουμε να μοιράσουμε κάποια ποσότητα σε N κόμβους, στη συγκεκριμένη περίπτωση το ρυθμό παραγωγής αιτήσεων, ώστε να ελαχιστοποιείται η αντικειμενική συνάρτηση. Η μέθοδος αυτή χρησιμοποιεί τις μερικές παραγώγους $\frac{\partial F_i}{\partial \lambda_{ij}}$ ώστε να λύσει το πρόβλημα.

Αφού ο χρήστης i θέλει να ελαχιστοποιήσει την αντικειμενική του συνάρτηση, θα προσπαθήσει να πάρει μια μικρή ποσότητα ρυθμού ε από τον εξυπηρετητή m και να το κατανείμει στον n έτσι ώστε η αντικειμενική του συνάρτηση να μειωθεί. Μάλιστα, με αυτή την ανακατανομή του ρυθμού ο i θα ήθελε να καταφέρει τη μεγαλύτερη δυνατή μείωση. Για να το καταφέρει αυτό, ο κόμβος i αφαιρεί ποσότητα ίση με ε από τον εξυπηρετητή m, με $m = \arg\max_{j \neq i} \frac{\partial F_i}{\partial \lambda_{ij}}$, και το κατανέμει στον n, με $n = \arg\min_{j \neq i} \frac{\partial F_i}{\partial \lambda_{ij}}$. Με άλλα λόγια, αφαιρείται ποσότητα ε από τον εξυπηρετητή που αντιστοιχεί στο μικρότερο ρυθμό μείωσης και προστίθεται σε εκείνον που θα προσφέρει τη μεγαλύτερη δυνατή μείωση στην αντικειμενική συνάρτηση. Η διαδικασία αύτη εξασφαλίζει ότι μειώνεται πάντα η αντικειμενική συνάρτηση.

Ο κόμβος i συνεχίζει τη διαδικασία ανακατανομής του ρυθμού παραγωγής αιτήσεων μέχρι να μην είναι δυνατή καμία περαιτέρω μείωση της αντικειμενικής συνάρτησης. Στο σημείο εκείνο, όλες οι μερικές παράγωγοι της αντικειμενικής συνάρτησης θα είναι ίσες μεταξύ τους και τότε θα έχει βρεθεί το βέλτιστο σημείο.

Τέλος, είναι σημαντικό να τονίσουμε τη σπουδαιότητα της ακρίβειας ε που χρησιμοποιούμε στον αλγόριθμο. Όσο πιο μεγάλη τιμή βάλουμε στο ε τόσο πιο γρήγορα θα συγκλίνει ο αλγόριθμος, αφού θα πλησιάζει γρηγορότερα στη βέλτιστη λύση, αλλά θα χάσουμε σε ακρίβεια. Αντίθετα, όταν επιλέξουμε μικρή τιμή για το ε ο αλγόριθμος θα χρειαστεί περισσότερες επαναλήψεις αλλά το αποτέλεσμα θα είναι πιο κοντά στην πραγματική βέλτιστη λύση.

Στη συνέχεια παρουσιάζουμε τους αλγορίθμους της προσέγγισης αυτής και δείχνουμε πως μπορούν να εφαρμοστούν σε κατανεμημένα πρωτόκολλα με την κατάλληλη ανταλλαγή μηνυμάτων ανάμεσα στους χρήστες. Για απλοποίηση των υπολογισμών από εδώ και στο εξής θεωρούμε ότι ο μέσος ρυθμός εξυπηρέτησης σε όλους τους εξυπηρετητές είναι $\mu = 1$. Επίσης, για ευκολία και χωρίς απώλεια της γενικότητας μπορούμε να θεωρήσουμε ότι οι κόμβοι ονοματίζονται βάσει της προτεραιότητας που απολαμβάνουν στο σύστημα, με τον πρώτο κόμβο να είναι αυτός με την μεγαλύτερη προτεραιότητα. Τέλος, θεωρούμε ότι κάθε αίτηση ενός κόμβου μπορεί να εξυπηρετηθεί από οποιοδήποτε άλλο κόμβο. Αυτό ισχύει στην περίπτωση που το δίκτυο αποτελείται μόνο από κόμβους που διαθέτουν ένα σημαντικό ποσοστό του αρχείου, τέτοιο ώστε κάθε άλλος κόμβος να έχει τη δυνατότητα να κατεβάσει τμήματα του από οποιονδήποτε άλλο.

Αλτρουιστική Προσέγγιση

Αυτή η προσέγγιση κατανεμημένης επίλυσης του προβλήματος ελαχιστοποίησης των χρόνων αναμονής των κόμβων βασίζεται στην ιδέα ότι κάθε κόμβος λύνει το πρόβλημα ελαχιστοποίησης για τον εαυτό του καθώς και για όλους τους κόμβους που έχουν μικρότερη προτεραιότητα από αυτόν. Με τον τρόπο αυτό, κάθε κόμβος λαμβάνει υπόψη του την επίδραση των αιτήσεών του στις καθυστερήσεις των κόμβων που επηρεάζει.

Στη συνέχεια θα παρουσιάσουμε δύο διαφορετικούς αλγορίθμους που βασίζονται στην αλτρουιστική προσέγγιση. Ο πρώτος είναι ένας επαναληπτικός αλγόριθμος, που μπορεί να λυθεί με πολύ μικρό υπολογιστικό κόστος χρησιμοποιώντας τη μέθοδο waterfilling και αποδεικνύεται ότι συγκλίνει στη βέλτιστη λύση του καθολικού προβλήματος, ενώ στον δεύτερο κάθε κόμβος προσπαθεί να λύσει ένα υποπρόβλημα του καθολικού προβλήματος. Ο δεύτερος αυτός αλγόριθμος συγκλίνει στη βέλτιστη λύση σε μία μόλις επανάληψη.

Altruistic Waterfilling

Περιγραφή Μεθόδου

Εφαρμόζοντας την αλτρουιστική προσέγγιση, κάθε κόμβος αποφασίζει πως θα στείλει τα αιτήματά του στους άλλους κόμβους λαμβάνοντας υπόψη και την επίδραση που θα έχει αυτό σε εκείνους με μικρότερη προτεραιότητα από αυτόν. Κάθε κόμβος ξεκινώντας από μια αρχική τυχαία ανάθεση ανακατανέμει μια μικρή ποσότητα ρυθμού μετάδοσης μεταξύ κάποιων δύο εξυπηρετητών, με τον τρόπο που περιγράψαμε και παραπάνω (Waterfilling), προσπαθώντας να επιτύχει μείωση στην αντικειμενική του συνάρτηση και επαναλαμβάνοντας μέχρι να μην είναι δυνατή περαιτέρω μείωσή της. Αφού όλοι οι κόμβοι επιλέξουν τα σπασίματα των ρυθμών μετάδοσής τους, η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε σε κάποιο σημείο ισορροπίας. Έτσι λοιπόν, σε κάθε επανάληψη κάθε κόμβος i, με σειρά προτεραιότητας, λύνει το παρακάτω πρόβλημα:

$$\min_{\lambda_{ij}} b_i \frac{\lambda_i}{\lambda} D_i + \sum_{k>i} b_k \frac{\lambda_k}{\lambda} D_k \tag{4.4}$$

$$\text{s.t} \quad \sum_{j \neq i} \lambda_{ij} = \lambda_i \quad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \quad \forall \text{j}$$

$$\lambda_{ij} \geq 0$$

Αντικαθιστώντας τις συνολικές καθυστερήσεις D_i και D_k των κόμβων με τις επιμέρους τιμές, καταλήγουμε στο παρακάτω πρόβλημα ελαχιστοποίησης για κάθε κόμβο i:

$$\min_{\lambda_{ij}} b_i \frac{\lambda_i}{\lambda} \sum_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij} + b_k \frac{\lambda_k}{\lambda} \sum_{k \neq j, k>i} \frac{\lambda_{kj}}{\lambda_k} D_{kj} \tag{4.5}$$

$$\text{s.t.} \quad \sum_{j=1, j \neq i}^{N} \lambda_{ij} = \lambda_i \quad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \quad \forall \text{j}$$

$$\lambda_{ij} > 0 \quad \forall i, j$$

με

$$D_{ij} = \frac{1}{\left(1 - \sum_{k \leq i} \lambda_{kj}\right)\left(1 - \sum_{k<i} \lambda_{kj}\right)} \tag{4.6}$$

Σημειώνουμε ότι και σε αυτή την περίπτωση, για λόγους ευκολίας, έχουμε κάνει την παραδοχή ότι $b_k > b_{k+1}$ και επομένως με τη συνθήκη $k > i$ στο άθροισμα εννοούμε όλους τους κόμβους με μικρότερη προτεραιότητα από τον i. Επίσης, αναφέρουμε ότι τα λ_{kj} των κόμβων με μικρότερη προτεραιότητα αποτελούν τιμές της προηγούμενης επανάληψης αφού σύμφωνα με τη σειρά επίλυσης του προβλήματος ελαχιστοποίησης, δεν έχουν ακόμη αποφασίσει για τα λ_{kj} τους σε αυτή την επανάληψη.

Το παραπάνω πρόβλημα αποτελεί ένα κυρτό πρόβλημα βελτιστοποίησης ως αθροίσματα κυρτών συναρτήσεων και αποδεικνύεται ότι η λύση του συγκλίνει στην βέλτιστη λύση του αντίστοιχου καθολικού προβλήματος που περιγράψαμε παραπάνω. Κάθε κόμβος i μπορεί να το λύσει εφαρμόζοντας κάποιο αλγόριθμο μερικών παραγώγων όπως το waterfilling, που περιγράφηκε αναλυτικά παραπάνω. Οι μερικές παράγωγοι της παραπάνω αντικειμενικής συνάρτησης F είναι:

$$\frac{\partial F}{\partial \lambda_{ij}} = b_i \left(D_{ij} + \frac{\lambda_{ij}}{\left(1-\sum_{k\le i}\lambda_{kj}\right)^2 \left(1-\sum_{k<i}\lambda_{kj}\right)} \right) + \sum_{q>1, q\neq j} b_q \lambda_{qj} \frac{2-\sum_{l<q}\lambda_{lj} - \sum_{l\le q}\lambda_{lj}}{\left(1-\sum_{l\le q}\lambda_{lj}\right)^2 \left(1-\sum_{l<q}\lambda_{lj}\right)^2} \qquad (4.7)$$

Για να μπορεί κάθε κόμβος να υπολογίσει τις παραπάνω μερικές παραγώγους χρειάζεται να έχει :

• Τις ποσότητες $\sum_{k<i}\lambda_{kj}$ για κάθε εξυπηρετητή j, που είναι απαραίτητες για τον πρώτο όρο της μερικής παραγώγου. Οι ποσότητες αυτές μπορούν να στέλνονται στον i είτε από τον αμέσως προηγούμενο κόμβο $i-1$ είτε από κάθε εξυπηρετητή j ξεχωριστά.

• Τις ποσότητες λ_{qj} της προηγούμενης επανάληψης των κόμβων με μικρότερη προτεραιότητα από τον i, ώστε να μπορεί να τις προσθέσει στα $\sum_{k<i}\lambda_{kj}$ και να παράγει τα αθροίσματα $\sum_{l\le q}\lambda_{lj}$ και $\sum_{l<q}\lambda_{lj}$. Και σε αυτή την περίπτωση αυτές οι ποσότητες μπορούν να σταλούν από τον αμέσως προηγούμενο κόμβο ή από κάθε εξυπηρετητή j ξεχωριστά.

Παράδειγμα εκτέλεσης

Για να γίνει αντιληπτή η διαδικασία με την οποία γίνεται η ανταλλαγή των πληροφοριών ώστε να υπολογιστούν οι μερικές παράγωγοι ας σκεφτούμε το παρακάτω απλό παράδειγμα. Ας υποθέσουμε ότι έχουμε ένα δίκτυο τεσσάρων κόμβων όπου ισχύει $b_1 < b_2 < b_3 < b_4$. Θα συμβολίζουμε με $\lambda_{ij}^{(t)}$ το ρυθμό με τον οποίο αποφάσισε να στέλνει αιτήματα ο κόμβος i στον εξυπηρετητή j κατά την επανάληψη t. Υποθέτουμε επίσης ότι βρισκόμαστε στην τρίτη επανάληψη και ότι τη χρονική στιγμή που παρατηρούμε το σύστημα ο κόμβος 2 θέλει να αποφασίσει ποια θα είναι τα $\lambda_{2j}^{(3)}$ που θα στείλει. Επομένως πρέπει να υπολογίσει, αρχικά, τις μερικές παραγώγους $\frac{\partial F}{\partial \lambda_{21}^{(3)}}, \frac{\partial F}{\partial \lambda_{23}^{(3)}}$ και $\frac{\partial F}{\partial \lambda_{24}^{(3)}}$. Θα υποθέσουμε επίσης ότι τις ποσότητες που χρειάζεται θα του τις προσφέρει κάθε εξυπηρετητής ξεχωριστά.

Αρχικά, δέχεται από τον εξυπηρετητή 1 το $\sum_{k<2}\lambda_{k1}^{(3)} = \lambda_{11}^{(3)}$, το οποίο είναι όμως ίσο με μηδέν αφού ο 1 δεν στέλνει τίποτα στον εαυτό του, και τα $\lambda_{31}^{(2)}$ και $\lambda_{41}^{(2)}$. Έτσι ο κόμβος 2, προσθέτοντας και το $\lambda_{21}^{(3)}$ που θέλει να δοκιμάσει, μπορεί τώρα να υπολογίσει τα $\sum_{k<2}\lambda_{k1}^{(3)} = 0$, $\sum_{k\le 2}\lambda_{k1}^{(3)} = \lambda_{21}^{(3)}$, $\sum_{l\le 3}\lambda_{l1} = \lambda_{21}^{(3)} + \lambda_{31}^{(2)}$, $\sum_{l\le 4}\lambda_{l1} = \lambda_{21}^{(3)} + \lambda_{31}^{(2)} + \lambda_{41}^{(2)}$ και τελικά να υπολογίσει την $\frac{\partial F}{\partial \lambda_{21}^{(3)}}$.

Στη συνέχεια, δέχεται από τον εξυπηρετητή 3 το $\sum_{k<2} \lambda_{k3}^{(3)} = \lambda_{13}^{(3)}$ και το $\lambda_{43}^{(2)}$. Με τις ποσότητες αυτές, ο 2 μπορεί να υπολογίσει τα $\sum_{k\le 2} \lambda_{k3}^{(3)} = \lambda_{13}^{(3)} + \lambda_{23}^{(3)}$, $\sum_{l\le 4} \lambda_{l3} = \lambda_{13}^{(3)} + \lambda_{23}^{(3)} + \lambda_{43}^{(2)}$ και τελικά να υπολογίσει την $\frac{\partial F}{\partial \lambda_{23}^{(3)}}$.

Έπειτα, ο εξυπηρετητής 4 στέλνει το $\sum_{k<2} \lambda_{k4}^{(3)} = \lambda_{14}^{(3)}$ και το $\lambda_{34}^{(2)}$. Στη συνέχεια, ο κόμβος 2 υπολογίζει το $\sum_{k\le 2} \lambda_{k4}^{(3)} = \lambda_{14}^{(3)} + \lambda_{24}^{(3)}$, το $\sum_{l\le 3} \lambda_{l4} = \lambda_{14}^{(3)} + \lambda_{24}^{(3)} + \lambda_{34}^{(2)}$ και τελικά την $\frac{\partial F}{\partial \lambda_{24}^{(3)}}$.

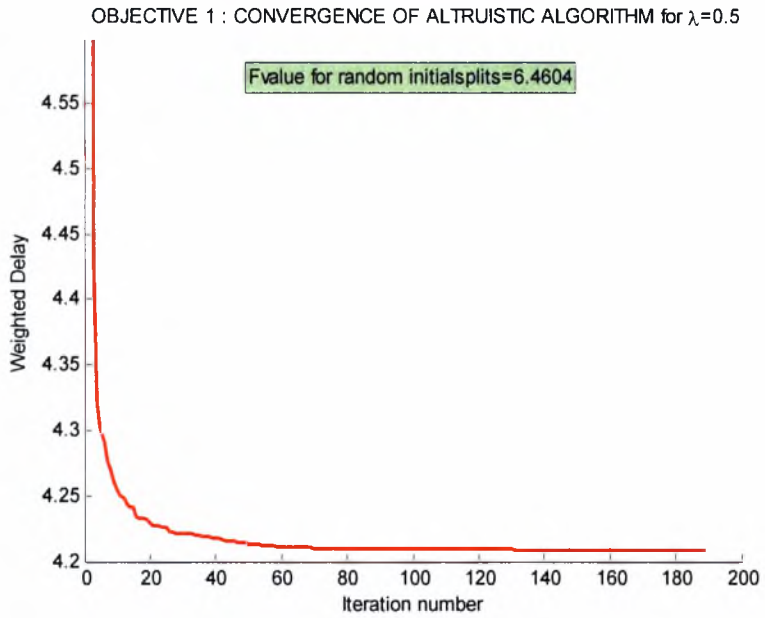
Τέλος, ο κόμβος 2 θα χρησιμοποιήσει τις τιμές αυτές των μερικών παραγώγων όπως ορίζει η μέθοδος Waterfilling για να υπολογίσει τα βέλτιστα $\lambda_{21}^{(3)}, \lambda_{23}^{(3)}$ και $\lambda_{24}^{(3)}$.

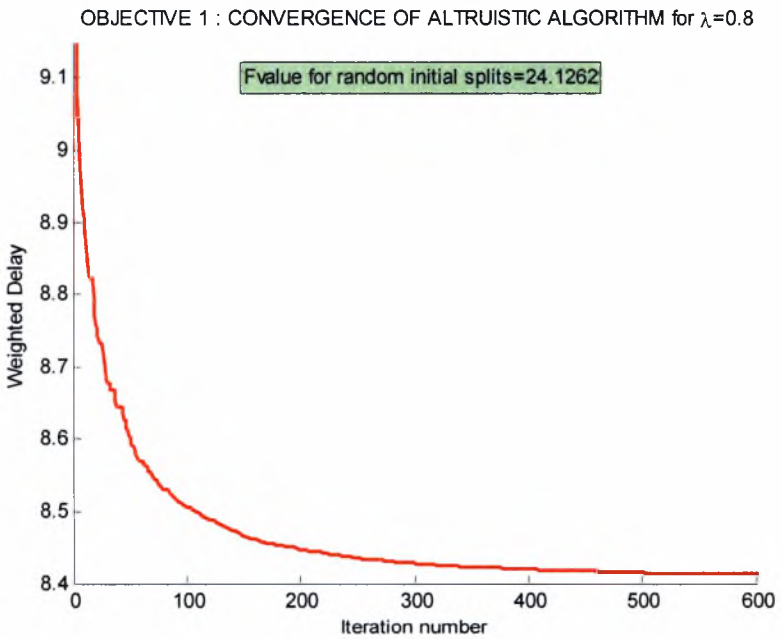
Προσομοίωση

Για την καλύτερη κατανόηση της συμπεριφοράς και την αξιολόγηση της απόδοσης των παραπάνω αλγορίθμων προχωρήσαμε στην υλοποίησή τους και στην προσομοίωση ενός δικτύου βάσει του μοντέλου που περιγράψαμε προηγουμένως. Το σύνολο των προσομοιώσεων πραγματοποιήθηκε σε περιβάλλον MATLAB και γενικά, εκτός αν αναφέρεται διαφορετικά, το δίκτυο που μοντελοποιούμε αποτελείται από N=4 κόμβους με διάνυσμα βαρών το **b**=[4 3 2 1]. Επίσης, όλοι οι κόμβοι έχουν τον ίδιο ρυθμό παραγωγής αιτήσεων λ_i και το βήμα της τεχνικής waterfilling έχει τιμή $\varepsilon = 10^{-3}$. Στη συνέχεια παρουσιάζουμε κάποια από τα αποτελέσματα που προέκυψαν.

I. Σύγκλιση αλγορίθμου

Στη συνέχεια, παρουσιάζουμε την ταχύτητα σύγκλισης του αλγορίθμου για δύο διαφορετικές περιπτώσεις ρυθμών αφίξεων λ για να δούμε πως συμπεριφέρεται ο αλγόριθμος για διαφορετικό φόρτο αιτήσεων. Συγκεκριμένα, απεικονίζουμε την τιμή της αντικειμενικής συνάρτησης μετά από κάθε εκτέλεση του αλγορίθμου από ένα κόμβο.





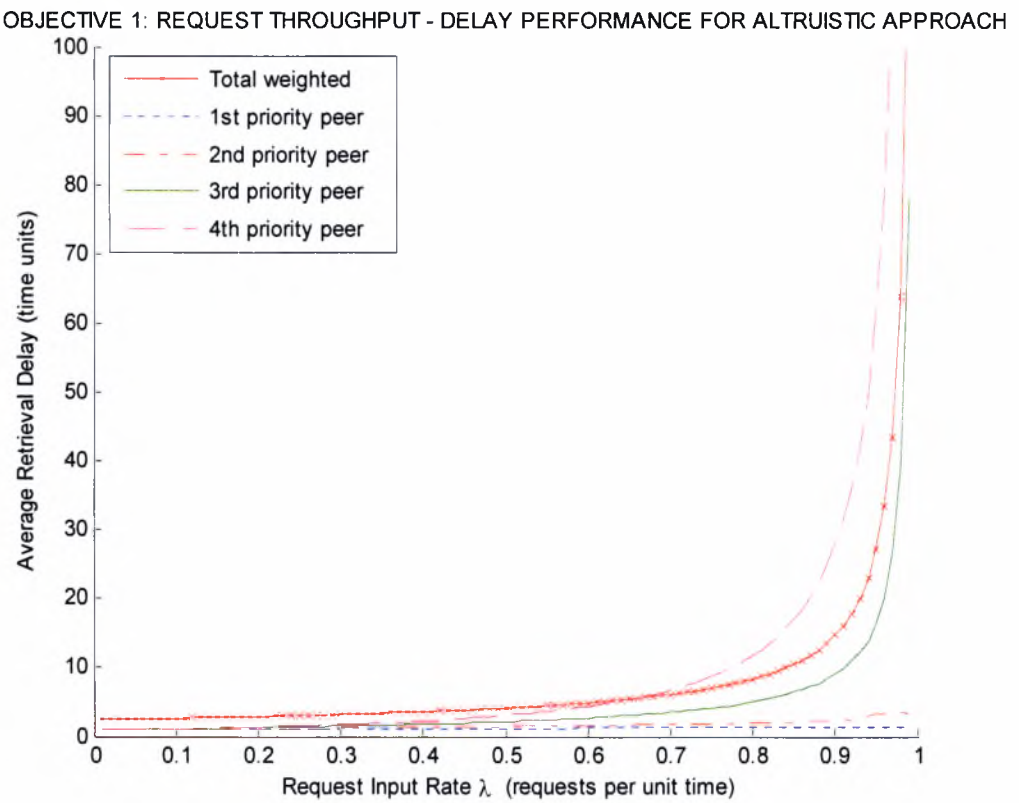
Αρχικά, είναι απαραίτητο να αναφέρουμε ότι εξαιτίας της τυχαίας αρχικής λύσης με την οποία ξεκινά ο αλγόριθμος, ο αριθμός των επαναλήψεων που απαιτούνται για την σύγκλιση δεν είναι σταθερός. Ωστόσο, στα παρακάτω γραφήματα παρουσιάζουμε μια μέση περίπτωση όπου μπορεί κανείς να βγάλει ασφαλή συμπεράσματα για την ταχύτητα σύγκλισης του αλγορίθμου. Επίσης, είναι αυτονόητο ότι καθώς ο αριθμός των κόμβων του συστήματος αυξάνεται απαιτούνται περισσότερες επαναλήψεις ώστε να επιτευχθεί η σύγκλιση.

Ακόμη, επιβεβαιώνουμε ότι η αντικειμενική συνάρτηση μειώνεται πάντα μεταξύ δύο επαναλήψεων γεγονός που άλλωστε αναμένεται λόγω του της μεθοδολογίας που ακολουθεί ο αλγόριθμος για να βρει τη βέλτιστη λύση. Για την περίπτωση που λ=0.8, παρατηρούμε ότι ο αλγόριθμος αν και χρειάζεται σχεδόν 500 επαναλήψεις για να συγκλίνει οριστικά, οι αλλαγές στην αντικειμενική συνάρτηση από την επανάληψη 300, περίπου, και μετά είναι ελάχιστες. Η ίδια συμπεριφορά παρατηρείται και στην

περίπτωση όπου το λ έχει την τιμή 0.5. Η αντικειμενική συνάρτηση ελαττώνεται ελάχιστα μετά την επανάληψη 70. Συγκρίνοντας την ταχύτητα σύγκλισης μεταξύ των δύο γραφημάτων παρατηρούμε ότι ο αλγόριθμος χρειάζεται περισσότερες επαναλήψεις για μεγάλο λ, κάτι που είναι λογικό αφού ο φόρτος του συστήματος είναι μεγαλύτερος με αποτέλεσμα ο αλγόριθμος να δυσκολεύεται περισσότερο να μοιράσει τους ρυθμούς αφίξεων με τον βέλτιστο τρόπο.

II. Απόδοση αλγορίθμου

Στο παρακάτω γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικό μέσος χρόνος ανάκτησης του συστήματος.



Ιδιαίτερο χαρακτηριστικό της γραφικής παράστασης είναι η μεγάλη αύξηση της καθυστέρησης καθώς αυξάνεται ο ρυθμός άφιξης αιτήσεων λ. Αυτό είναι απολύτως φυσιολογικό αφού ο φόρτος του συστήματος αυξάνεται καθώς μεγαλώνει το λ κάθε κόμβου, προκαλώντας μεγαλύτερη καθυστέρηση, και ιδιαίτερα όταν πλησιάζουμε στο ρυθμό εξυπηρέτησης C, όπου το σύστημα αγγίζει τα όρια ευστάθειάς του. Ένα άλλο σημαντικό χαρακτηριστικό του συστήματος, το οποίο έχουμε την ευκαιρία να επιβεβαιώσουμε με αυτή τη γραφική, είναι η πολύ μεγάλη καθυστέρηση που αντιμετωπίζει ο κόμβος 4, εκείνος δηλαδή με τη μικρότερη προτεραιότητα. Αυτό συμβαίνει γιατί ο αλγόριθμος δίνει πολύ μεγάλη βαρύτητα στην ελαχιστοποίηση του χρόνου ανάκτησης των κόμβων με μεγάλη προτεραιότητα, ιδιαίτερα του κόμβου 1, γεγονός που λειτουργεί αρνητικά για την καθυστέρηση εκείνων με μικρότερη προτεραιότητα, και ιδιαίτερα του κόμβου 4. Άλλωστε, είναι εύκολο να αντιληφθούμε

ότι ο κόμβος 1 έχοντας απόλυτη προτεραιότητα επηρεάζεται μόνο από την αύξηση του δικού του ρυθμού παραγωγής αιτήσεων, ενώ ο κόμβος 4 από την αύξηση όλων των λ_i. Ως επιβεβαίωση των παραπάνω, παρατηρούμε ότι οι κόμβοι με μεγάλη προτεραιότητα, δηλαδή οι 1 και 2, έχουν σχεδόν μηδενική καθυστέρηση (ο κόμβος 1 έχει, όπως ήταν φυσιολογικό, μικρότερη καθυστέρηση από τον 2) ενώ οι κόμβοι 3 και 4 έχουν αρκετά μεγαλύτερη. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα.

D_1	D_2	D_3	D_4	Total Delay
1.449	2.426	9.049	27.94	14.78

Τέλος, η συνολική καθυστέρηση στο σύστημα, όντας ο βεβαρημένος μέσος όρος των καθυστερήσεων με βάρη μεγαλύτερα της μονάδας, θα περιμέναμε να βρίσκεται πάνω από τις επιμέρους καθυστερήσεις των κόμβων. Αυτό γενικά ισχύει, όχι όμως και για μεγάλες τιμές του λ_i. Σε αυτές τις περιπτώσεις η τιμή της συνολικής καθυστέρησης είναι μικρότερη από αυτή του κόμβου 4 λόγω της ραγδαίας αύξησης που περιγράψαμε παραπάνω.

Altruistic Subglobal

Περιγραφή Μεθόδου

Ο αλγόριθμος αυτός είναι ένας μη επαναληπτικός αλγόριθμος που βασίζεται στην αλτρουιστική προσέγγιση που αναφέραμε πιο πάνω. Κάθε κόμβος i με σειρά προτεραιότητας λύνει ένα υποπρόβλημα του καθολικού προβλήματος, που αναφέρεται σε αυτόν και όλους τους κόμβους που έχουν μικρότερη προτεραιότητα από αυτόν. Έτσι λοιπόν, αν έχουμε ένα δίκτυο από N κόμβους, ο κόμβος i λύνει το παρακάτω πρόβλημα ελαχιστοποίησης:

$$\min_{\lambda_{kj}} \sum_{k=i}^{N} b_k \frac{\lambda_k}{\lambda} \sum_{j \neq k} \frac{\lambda_{kj}}{\lambda_k} D_{kj} \tag{4.8}$$

$$\text{s.t} \quad \sum_{j \neq k} \lambda_{kj} = \lambda_k \qquad \forall k$$

$$\sum_{k=1}^{N} \lambda_{kj} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} \geq 0$$

όπου το D_{kj} δίνεται από τη σχέση (4.6).

Το παραπάνω πρόβλημα είναι ένα πρόβλημα ελαχιστοποίησης με $(N-i)N$ μεταβλητές που μπορεί να λυθεί με μεθόδους βασισμένες στη χρήση μερικών παραγώγων, όχι όμως και από τη μέθοδο Waterfilling. Κάθε κόμβος i αφού λύσει το πρόβλημα κρατάει μόνο το τμήμα της λύσης που αναφέρεται σε αυτόν, δηλαδή την i-στη γραμμή της λύσης, αγνοώντας για τις υπόλοιπες. Με μια πιο προσεκτική εξέταση του αλγορίθμου είναι εύκολο κανείς να διαπιστώσει ότι ο πρώτος κόμβος

λύνει στην πραγματικότητα το καθολικό πρόβλημα ενώ ο τελευταίος, δηλαδή εκείνος με την μικρότερη προτεραιότητα, λύνει το εγωιστικό πρόβλημα.

Οι μερικές παράγωγοι της αντικειμενικής συνάρτησης του προβλήματος (4.8) είναι ίδιες με αυτές του προηγούμενου αλτρουιστικού αλγορίθμου, αυτές δηλαδή που ορίζονται από τη σχέση (4.7). Για να μπορεί κάθε κόμβος να λύσει το πρόβλημα που του αντιστοιχεί χρειάζεται να έχει τις ποσότητες $\sum_{k<i} \lambda_{kj}$, δηλαδή το άθροισμα των ρυθμών των κόμβων με μεγαλύτερη προτεραιότητα. Όπως και πριν, αυτές οι ποσότητες μπορούν να στέλνονται στον κόμβο i είτε από τον αμέσως προηγούμενο κόμβο είτε από κάθε εξυπηρετητή j ξεχωριστά.

Παράδειγμα εκτέλεσης

Για να γίνουν εμφανείς οι ανταλλαγές των πληροφοριών που απαιτούνται ας σκεφτούμε και πάλι το πολύ απλό παράδειγμα του δικτύου των τεσσάρων κόμβων και ας υποθέσουμε ότι είναι και πάλι η σειρά του κόμβου 2 να λύσει το πρόβλημα που του αντιστοιχεί, δηλαδή το πρόβλημα:

$$\min_{\lambda_{kj}} \sum_{k=2}^{N} b_k \frac{\lambda_k}{\lambda} \sum_{j \neq k} \frac{\lambda_{kj}}{\lambda_k} D_{kj} \qquad (4.9)$$

$$\text{s.t} \quad \sum_{j \neq k} \lambda_{kj} = \lambda_k \qquad \forall k$$

$$\sum_{k=1}^{N} \lambda_{kj} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} \geq 0$$

Επίσης, υποθέτουμε ότι τις ποσότητες που απαιτούνται θα τις πάρει από κάθε εξυπηρετητή j ξεχωριστά.

Αρχικά, παίρνει από τους εξυπηρετητές 3 και 4 τα $\sum_{k<2} \lambda_{k3} = \lambda_{13}$ και $\sum_{k<2} \lambda_{k4} = \lambda_{14}$ αντίστοιχα. Από τον εξυπηρετητή 1 δεν παίρνει τίποτα αφού δεν έχει στείλει ακόμη κανείς τίποτα γιατί ο 2 είναι ο κόμβος με τη μεγαλύτερη προτεραιότητα στον εξυπηρετητή 1. Χρησιμοποιώντας αυτές τις ποσότητες λύνει το πρόβλημα (4.9) με κάποια από τις μεθόδους που αναφέραμε παραπάνω. Η λύση είναι ένας πίνακας τριών γραμμών και τεσσάρων στηλών που αντιστοιχούν στα σπασίματα των κόμβων 2, 3 και 4 αντίστοιχα. Από αυτές τις γραμμές κρατάει μόνο την πρώτη σειρά, η οποία περιέχει τα σπασίματα του δικού του ρυθμού παραγωγής αιτήσεων, τα στοιχεία της οποίας κοινοποιεί στους αντίστοιχους εξυπηρετητές.

Εξετάζοντας τη λογική που διέπει τον αλγόριθμο βλέπουμε ότι ο πρώτος κόμβος λύνοντας το πρόβλημα που του αντιστοιχεί θα επιλέξει τα ίδια λ_{1j} που ορίζονται και από τη λύση του καθολικού προβλήματος, αφού στην ουσία λύνει αυτό το πρόβλημα. Ο δεύτερος, έχοντας ως δεδομένη την επιλογή του πρώτου, θα οδηγηθεί στο διάνυσμα λ_{2j} του καθολικού προβλήματος, και γενικά όταν ένας κόμβος k πρέπει να λύσει το πρόβλημα που του αντιστοιχεί, έχοντας ως δεδομένες τις λύσεις των άλλων, θα βρίσκει ένα διάνυσμα λ_{kj} που θα είναι ακριβώς ίδιο με το αντίστοιχο τμήμα της λύσης του καθολικού προβλήματος.

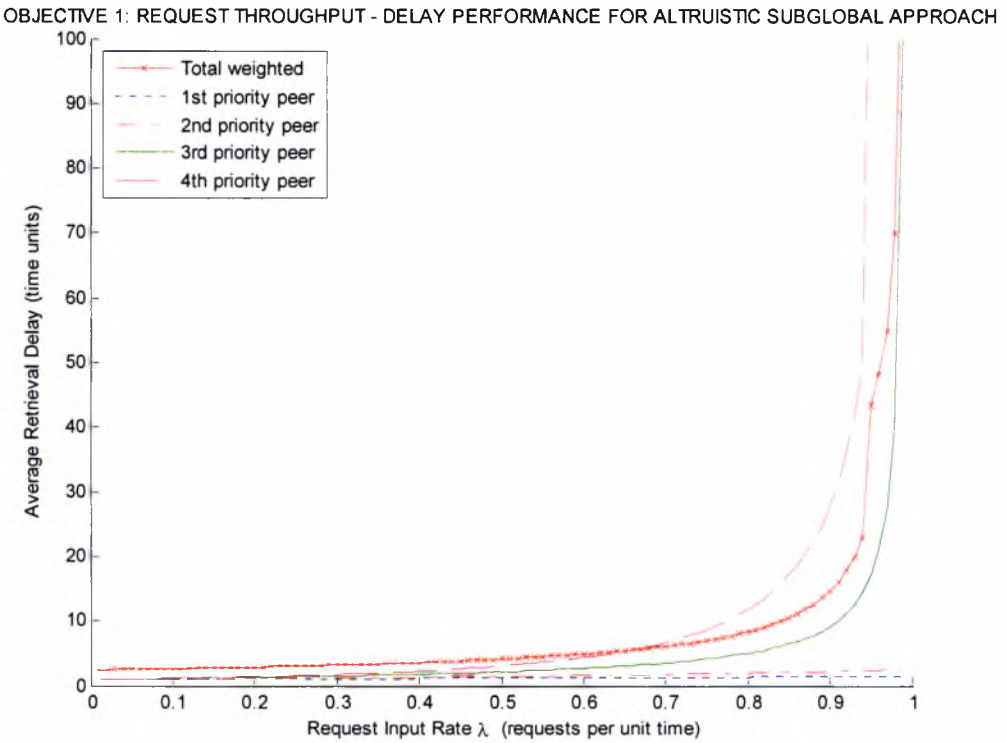
Συμπερασματικά, πρόκειται για έναν κατανεμημένο αλγόριθμο που σε μία μόνο επανάληψη και με μικρό κόστος σε ανταλλαγή μηνυμάτων μπορεί να λύσει με τον καλύτερο τρόπο το πρόβλημα της ελαχιστοποίησης του συνολικού χρόνου

εξυπηρέτησης μιας αίτησης αφού στην ουσία λύνει το καθολικό πρόβλημα με κατανεμημένο τρόπο.

Προσομοίωση

Απόδοση αλγορίθμου

Όπως είπαμε και παραπάνω, ο αλγόριθμος αυτός λύνει στην ουσία το καθολικό πρόβλημα ελαχιστοποίησης αλλά με κατανεμημένο τρόπο. Στο παρακάτω γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός μέσος χρόνος ανάκτησης του συστήματος.



Συγκρίνοντας το παραπάνω διάγραμμα με αυτό του προηγούμενου αλτρουιστικού αλγορίθμου βλέπουμε ότι είναι ίδια. Αυτό είναι απολύτως φυσιολογικό αφού, όπως είπαμε παραπάνω, η επαναληπτική αλτρουιστική μέθοδος αποδεικνύεται ότι συγκλίνει στη λύση του καθολικού προβλήματος. Επομένως, οι παρατηρήσεις που κάναμε σχετικά με τη συμπεριφορά των καθυστερήσεων των κόμβων ισχύουν και εδώ με την διαφορά ότι στην περίπτωση δεν μπορεί να υπάρξει διάγραμμα σύγκλισης αφού ο αλγόριθμος δεν είναι επαναληπτικός. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα:

D_1	D_2	D_3	D_4	Total Delay
1.452	2.336	9.187	27.87	14.76

Συγκρίνοντας τον παραπάνω πίνακα με τον αντίστοιχο του προηγούμενου αλγορίθμου βλέπουμε ότι οι καθυστερήσεις διαφέρουν ελάχιστα. Οι όποιες μικρές διαφορές οφείλονται στην ακρίβεια των μεθόδων επίλυσης που χρησιμοποιούνται.

Τέλος, αν θέλαμε να συγκρίνουμε τους δύο αλγορίθμους θα μπορούσαμε να πούμε ότι ο πρώτος μας προσφέρει εύκολη υλοποίηση, αφού υλοποιείται με τον αλγόριθμο waterfilling, ο οποίος έχει μικρές απαιτήσεις σε επεξεργαστική ισχύ, αλλά έχει το μειονέκτημα ότι χρειάζεται κάποιο αριθμό επαναλήψεων για να φτάσει στη βέλτιστη λύση. Αντίθετα, ο δεύτερος αλγόριθμος προσφέρεται για περιπτώσεις όπου χρειαζόμαστε στιγμιαία σύγκλιση, όμως απαιτεί την επίλυση προβλημάτων ελαχιστοποίησης πολλών μεταβλητών, αφού κάθε κόμβος λύνει ένα υποπρόβλημα του καθολικού, το οποίο για την περίπτωση του κόμβου με τη μεγαλύτερη προτεραιότητα ανάγεται στο ίδιο το καθολικό πρόβλημα. Αυτό έχει ως αποτέλεσμα να απαιτείται περισσότερος χρόνος για την επίλυσή του.

Packet – oriented Προσομοίωση Δικτύου

Οι αλγόριθμοι που παρουσιάσαμε μέχρι τώρα έχουν σαν κοινό χαρακτηριστικό ότι οι προτεραιότητες εξυπηρέτησης είναι προκαθορισμένες, ίδιες για όλους τους κόμβους και σταθερές σε όλη τη διάρκεια της προσομοίωσης. Στη συνέχεια θα παρουσιάσουμε έναν ευρεστικό αλγόριθμο ελαχιστοποίησης της καθυστέρησης όπου οι προτεραιότητες δεν είναι καθολικές, αλλά διαμορφώνονται δυναμικά με βάση το ρυθμό εξυπηρέτησης κάθε κόμβου, ο οποίος μπορεί να είναι διαφορετικός για κάθε εξυπηρετητή. Ο αλγόριθμος αυτός θα χρησιμοποιηθεί για να διαπιστώσουμε την επίδραση της εγωιστικής συμπεριφοράς στον καθορισμό των προτεραιοτήτων.

Το μοντέλο της προσομοίωσης

Το μοντέλο που προσομοιώνουμε αποτελείται από ένα δίκτυο τριών κόμβων. Καθένας από τους κόμβους λειτουργεί ως *πελάτης (client)* και ως *εξυπηρετητής (server)* σε ένα δίκτυο διαμοιρασμού αρχείων. Ως, πελάτης, κάθε κόμβος ζητά, σε τυχαίες χρονικές στιγμές, αρχεία από τους άλλους δύο, ενώ ως εξυπηρετητής ικανοποιεί τις αιτήσεις των άλλων κόμβων, που φτάνουν σε αυτόν, για αρχεία. Κάθε κόμβος μπορεί να έχει διαφορετικό ρυθμό παραγωγής αιτήσεων και ρυθμό εξυπηρέτησης από τους υπόλοιπους φροντίζοντας όμως πάντα να υπάρχει ευστάθεια σε κάθε κόμβο και επομένως και συνολικά στο δίκτυο, δηλαδή σε κανένα κόμβο να μην υπάρχει συνολικός ρυθμός άφιξης αιτήσεων μεγαλύτερος από το αντίστοιχο ρυθμό εξυπηρέτησης. Επίσης, είναι απαραίτητο να αναφέρουμε ότι πρόκειται για ένα *non-preemptive μοντέλο εξυπηρέτησης* δηλαδή όταν μια αίτηση αρχίσει να εκτελείται δεν πρόκειται να διακοπεί ακόμη και αν έρθει στην ουρά μια αίτηση με μεγαλύτερη προτεραιότητα.

Συμπεριφορά κόμβου ως πελάτης – client

Ο κόμβος i γεννά αιτήσεις που απευθύνονται στους άλλους κόμβους με ρυθμό λ_i. Οι χρονικές στιγμές της γέννησης των αιτήσεων αυτών ακολουθούν κατανομή *Poisson* και επομένως τα διαστήματα μεταξύ των αιτήσεων ακολουθούν την εκθετική κατανομή. Ο συνολικός ρυθμός παραγωγής αιτήσεων λ_i για τον κόμβο i, χωρίζεται σε δύο επιμέρους ρυθμούς, τον ρυθμό με τον οποίο θα στέλνονται οι αιτήσεις στον ένα γείτονα, και στον ρυθμό που θα στέλνονται στον άλλο. Για παράδειγμα, ο κόμβος 1 παράγει αιτήσεις με ρυθμό λ_1, από τις οποίες κάποιες, με ρυθμό λ_{12}, στέλνονται στον κόμβο 2 και με λ_{13} στον κόμβο 3. Δηλαδή σε κάθε στιγμή για τον κόμβο 1 ισχύει: $\lambda_1 = \lambda_{12} + \lambda_{13}$. Ο τρόπος με τον οποίο γίνεται το «σπάσιμο» του ρυθμού παραγωγής των αιτήσεων και συνεπώς η απόφαση για τον κόμβο στον οποίο θα σταλεί η κάθε αίτηση θα εξηγηθεί στη συνέχεια.

Μετά από τη γέννηση μιας αίτησης, ο κόμβος πρέπει να αποφασίσει σε ποιον θα στείλει την αίτηση. Ο κόμβος αποφασίζει τυχαία τον κόμβο στον οποίο θα στείλει την αίτηση με πιθανότητα όμως μεγαλύτερη να την στείλει στον κόμβο για τον οποίο έχει καλύτερη γνώμη. Η άποψη ενός κόμβου για τους γείτονές του διαμορφώνεται με βάση τη συμπεριφορά τους στα αιτήματά του στη διάρκεια των προηγούμενων εποχών. Έτσι, ο γείτονας που επέφερε μικρότερη μέση καθυστέρηση στην εξυπηρέτηση των αιτήσεών του θα είναι και αυτός για τον οποίο έχει καλύτερη γνώμη.

Συμπεριφορά κόμβου ως εξυπηρετητής - server

Κάθε κόμβος χαρακτηρίζεται από το ρυθμό εξυπηρέτησης, ο οποίος στο πρόβλημα που μελετάμε αντικατοπτρίζει τη χωρητικότητα της γραμμής με την οποία συνδέεται στο δίκτυο. Επιπλέον, σε κάθε κόμβο φτάνουν αιτήσεις που προέρχονται από τους δύο γείτονές του. Το γεγονός αυτό καθιστά απαραίτητη τη χρησιμοποίηση κάποιου αλγορίθμου εξυπηρέτησης των αιτήσεων. Υπάρχουν διάφοροι τρόποι εξυπηρέτησης, καθένας από τους οποίους δίνει διαφορετικά αποτελέσματα όσον αφορά την καθυστέρηση κάθε κόμβου ξεχωριστά και τη συνολική καθυστέρηση του συστήματος. Οι μέθοδοι καθορισμού της σειράς εξυπηρέτησης που θα εξετάσουμε είναι οι: *First Come First Served*, *Shortest Job First* καθώς και μία μέθοδος που χρησιμοποιεί ένα δυναμικό τρόπο καθορισμού των προτεραιοτήτων εξυπηρέτησης. Όλες αυτές οι μέθοδοι θα εξηγηθούν στη συνέχεια.

Η δομή της προσομοίωσης

Η προσομοίωση είναι οργανωμένη σε *εποχές (epochs)*. Σε κάθε μία τέτοια εποχή προσομοιώνουμε το δίκτυο με τους διάφορους τρόπους εξυπηρέτησης για ένα συγκεκριμένο χρονικό διάστημα. Κατά τη διάρκεια μια εποχής, κάθε κόμβος, ανάλογα με το ρυθμό λ που έχει, παράγει έναν αριθμό αιτήσεων.

Ως πελάτης, κάθε κόμβος αποφασίζει σε ποιον από τους δύο γείτονες θα στείλει κάθε αίτηση ανάλογα με τη γνώμη του για αυτούς. Η γνώμη κάθε κόμβου για τους άλλους διαμορφώνεται βάσει της μέσης καθυστέρησης που είχαν οι αιτήσεις του σε αυτούς στο σύνολο των προηγούμενων εποχών. Συγκεκριμένα, αν $wait(1,2)$ και

$wait(1,3)$ η μέση καθυστέρηση του κόμβου 1 στους 2 και 3 αντίστοιχα, η γνώμη του διαμορφώνεται βάσει των παρακάτω τύπων:

$$reputation(1,2) = \frac{wait(1,3)}{wait(1,2) + wait(1,3)} \tag{4.10}$$

και

$$reputation(1,3) = \frac{wait(1,2)}{wait(1,2) + wait(1,3)} \tag{4.11}$$

Επίσης, κάθε κόμβος εξυπηρετεί τις αιτήσεις που καταφθάνουν σε αυτόν με κάποια μέθοδο από τις παρακάτω.

First Come – First Served

Κάθε αίτηση εξυπηρετείται με βάση τη σειρά άφιξης στην ουρά. Η αίτηση που ήρθε πρώτη εξυπηρετείται και πρώτη. Αυτή η μέθοδος δίνει περισσότερο έμφαση στη δικαιοσύνη εξυπηρέτησης και λιγότερο στην αποδοτική εξυπηρέτηση των αιτήσεων. Αυτό συμβαίνει γιατί αιτήματα που απαιτούν πολύ χρόνο για να ικανοποιηθούν θα καθυστερούν υπερβολικά τις μικρότερες, ειδικά όταν οι χρόνοι εξυπηρέτησης των αιτήσεων έχουν μεγάλη διασπορά. Επιπλέον, αποφεύγεται ο κίνδυνος λιμοκτονίας.

Shortest Job First

Κάθε χρονική στιγμή εξυπηρετείται η αίτηση με το μικρότερο χρόνο εξυπηρέτησης. Με τον τρόπο αυτό οι μικρότερες αιτήσεις παραγκωνίζουν τις μεγαλύτερες με κίνδυνο να υπάρξει λιμοκτονία. Ωστόσο, οι μικρότερες αιτήσεις δεν είναι υποχρεωμένες να περιμένουν πολύ χρόνο πίσω από άλλες με πολύ μεγάλο χρόνο εξυπηρέτησης. Η μέθοδος αυτή, όπως είναι εμφανές, δίνει έμφαση στην αποδοτική ικανοποίηση των αιτημάτων και έχει αποδειχθεί ότι συντελεί στη ελαχιστοποίηση του μέσου χρόνο αναμονής.

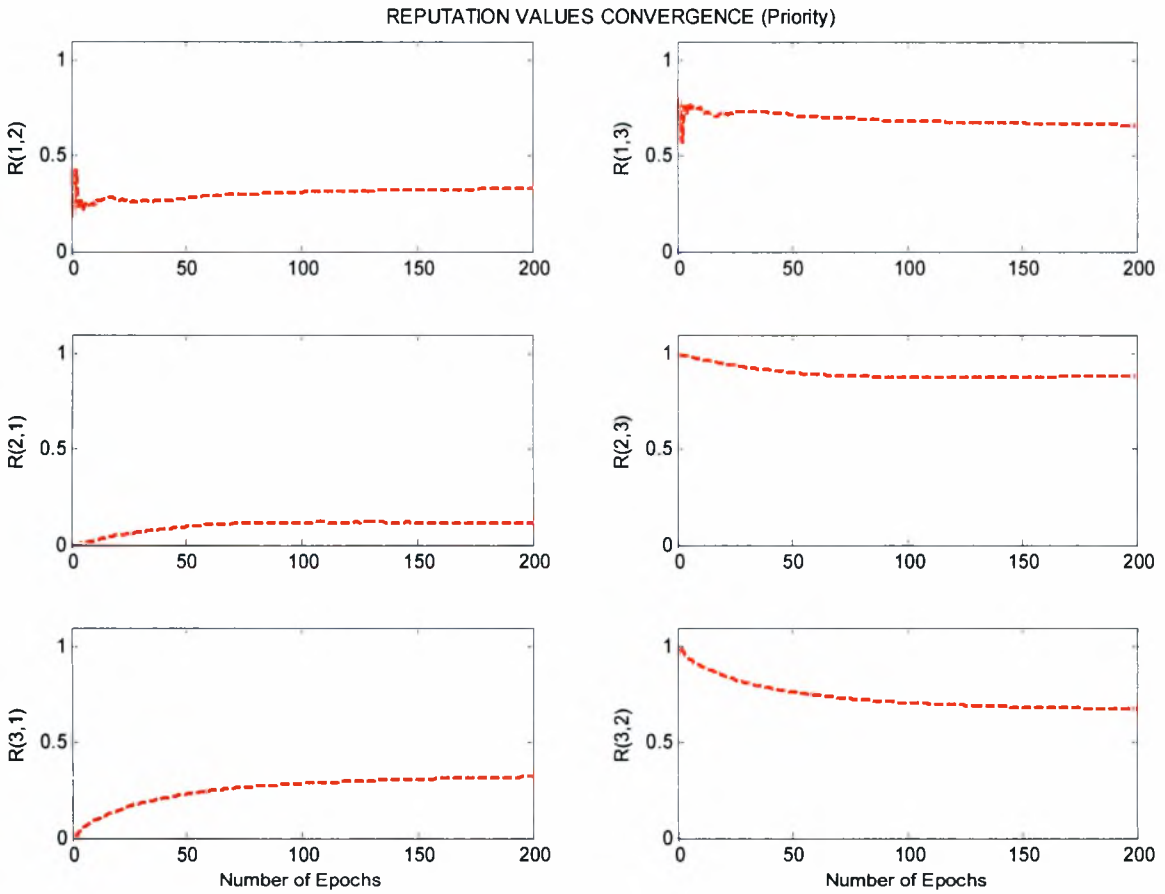
Μέθοδος εξυπηρέτησης με δυναμικό καθορισμό προτεραιοτήτων

Πρόκειται για μια μέθοδο εξυπηρέτησης όπου κάθε κόμβος έχει διαφορετική προτεραιότητα. Οι προτεραιότητες καθορίζονται στο τέλος κάθε εποχής, βάσει της τιμής reputation. Συγκεκριμένα, προτεραιότητα δίνεται στον κόμβο με το μεγαλύτερο reputation. Αυτός είναι και ένας τρόπος επιβράβευσης του γείτονα για την καλύτερη εξυπηρέτηση που του προσέφερε αφού έτσι θα εξυπηρετούνται με πολύ μικρή καθυστέρηση οι αιτήσεις του. Αυτή βέβαια είναι και μια αιτία δημιουργίας *Συμμαχιών (coalitions)* μεταξύ κάποιων κόμβων και εις βάρος κάποιων άλλων, φαινόμενο που θα αναλύσουμε και παρακάτω.

Προσομοίωση

Ι. Σύγκλιση αλγορίθμου

Όπως αναφέραμε και στον ορισμό του ευρεστικού αυτού αλγορίθμου ο κάθε κόμβος επιλέγει τα σπασίματά του βάσει της καθυστέρησης που αντιμετώπισε κατά την εξυπηρέτηση του από τους άλλους κόμβους τις προηγούμενες εποχές. Έτσι λοιπόν στο παρακάτω γράφημα παρουσιάζουμε την σύγκλιση των τιμών reputation για το δίκτυο των τριών κόμβων με μ=[0.5 1 1.5].



Παρατηρούμε ότι μετά από ένα αριθμό από εποχές οι τιμές reputation σταθεροποιούνται για όλους τους κόμβους. Ειδικότερα μπορούμε να δούμε ότι κάθε κόμβος στέλνει το μεγαλύτερο μέρος των αιτήσεων του στον κόμβο με τον μεγαλύτερο ρυθμό εξυπηρέτησης. Ενδεικτικά παραθέτουμε στον παρακάτω πίνακα τις τιμές του Reputation όταν έχει επιτευχθεί η σύγκλιση

Client \ Server	1	2	3
1	0	0.3361	0.6639
2	0.1125	0	0.8875
3	0.3221	0.6779	0

Όπως παρατηρούμε και στον παραπάνω πίνακα οι κόμβοι 2 και 3 είναι αυτοί που δέχονται το μεγαλύτερο μέρος των αιτήσεων και επίσης ο ένας δίνει στον άλλο μεγάλη τιμή reputation και συνεπώς και προτεραιότητα. Αυτό είναι ενδεικτικό της δημιουργίας συνέργιας ανάμεσα στους δύο κόμβους γεγονός που θα περιγράψουμε αναλυτικότερα και στη συνέχεια.

II. Απόδοση αλγορίθμου

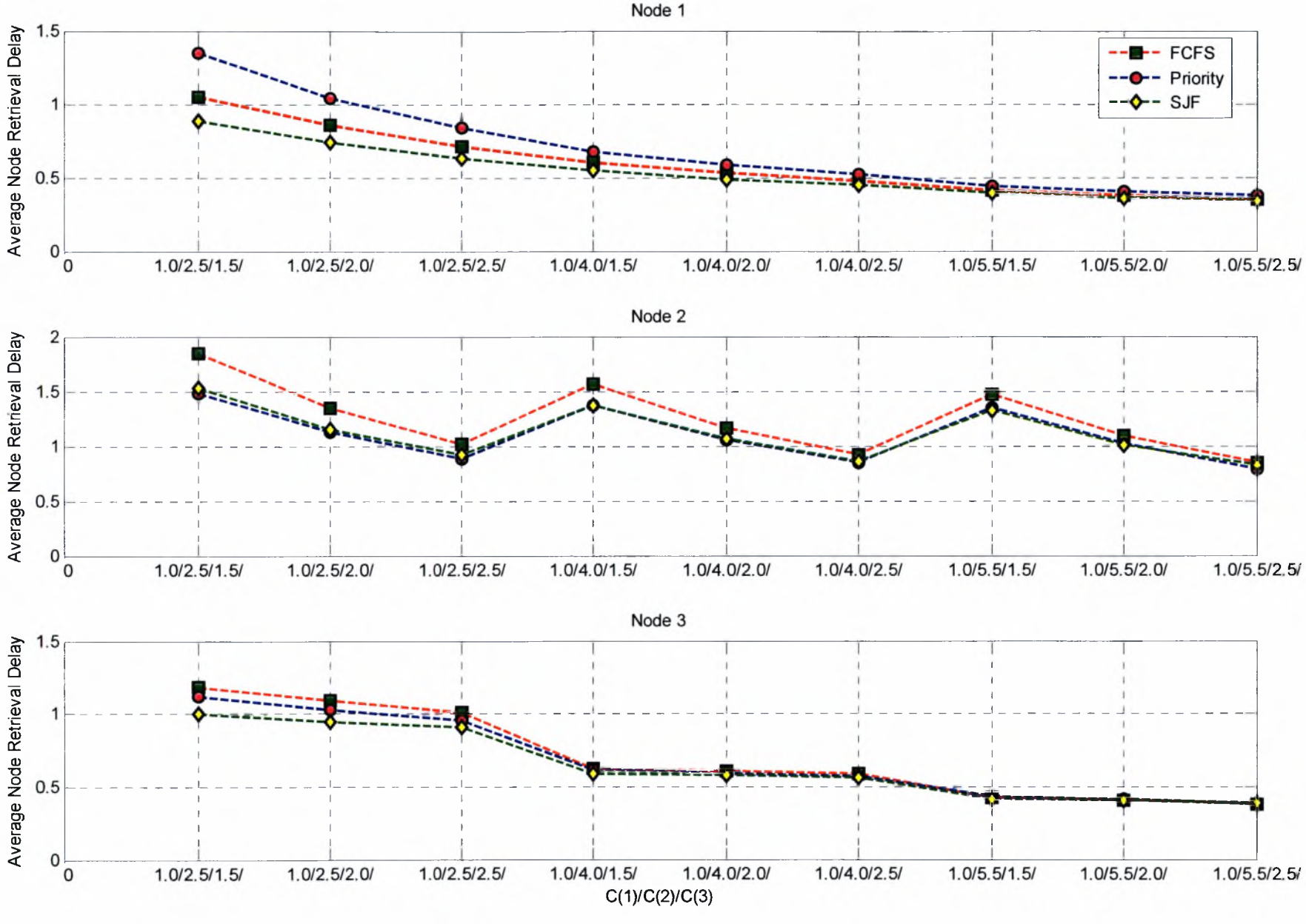
Για τη μέτρηση της απόδοσης του αλγορίθμου δυναμικών προτεραιοτήτων προσομοιώσαμε το σύστημα των τριών κόμβων για διάφορα διανύσματα C και συγκρίναμε τα αποτελέσματά του με εκείνα των μεθόδων εξυπηρέτησης *First Come First Served* και *Shortest Job First*. Εξαιτίας της διαφορετικής απόδοσης του αλγορίθμου για κάθε κόμβο λόγω της εμφάνισης συμμαχιών, παρουσιάζουμε παρακάτω τα διαγράμματα απόδοσης για κάθε κόμβο του συστήματος ξεχωριστά. Κατά τη διάρκεια της προσομοίωσης κρατήσαμε το ρυθμό εξυπηρέτησης $C(1)$ σταθερό και πάντα μικρότερο από τους άλλους δύο ρυθμούς ώστε να παρατηρήσουμε την εμφάνιση συμμαχίας μεταξύ των κόμβων 2 και 3.

Η πρώτη παρατήρηση που μπορούμε να κάνουμε αφορά τις κλίσεις των γραφικών, οι οποίες έχουν πτωτική πορεία εξαιτίας της συνεχόμενης αύξησης του συνολικού ρυθμού εξυπηρέτησης από πείραμα σε πείραμα. Όσον αφορά τον κόμβο 1, παρατηρούμε ότι σε κάθε περίπτωση ο αλγόριθμος προτεραιοτήτων του προκαλεί μεγαλύτερη καθυστέρηση σε σχέση με της άλλες μεθόδους, φαινόμενο που είναι απολύτως φυσιολογικό αφού ο 1 δε συμμετέχει σε καμία συμμαχία και επομένως τα αιτήματά του δεν έχουν προτεραιότητα σε καμία ουρά. Επιπλέον, επιβεβαιώνουμε ότι ο αλγόριθμος SJF έχει καλύτερα αποτελέσματα από τους άλλους δύο για τον κόμβο 1, αφού, όπως γνωρίζουμε από τη θεωρεία, ο καλύτερος τρόπος εξυπηρέτησης, όταν γνωρίζουμε το χρόνος εκτέλεσης των αιτημάτων, είναι κατά αύξοντα χρόνο εξυπηρέτησης.

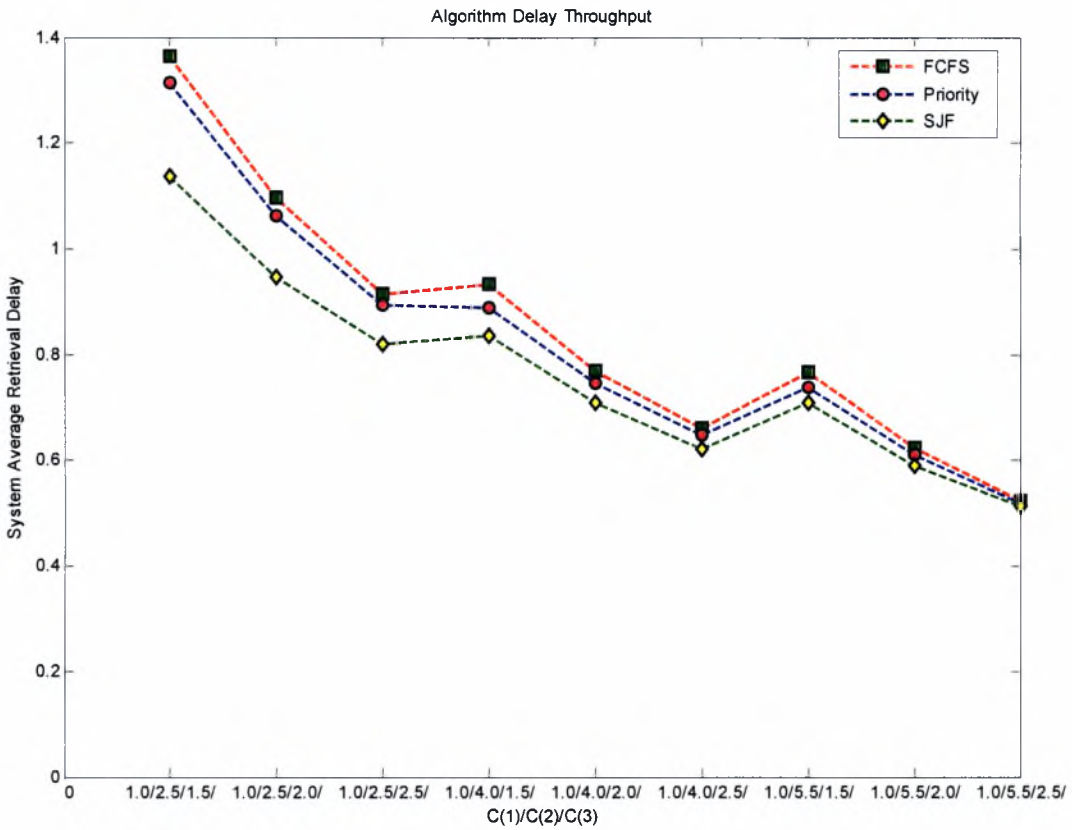
Για τους άλλους δύο κόμβους, όμως, τα συμπεράσματα είναι λίγο διαφορετικά. Οι κόμβοι αυτοί έχουν δημιουργήσει συμμαχία σε όλα τα πειράματα με αποτέλεσμα με τον αλγόριθμο προτεραιοτήτων να έχουν πολύ μικρές καθυστερήσεις, που σε ορισμένες περιπτώσεις είναι λίγο μικρότερες ακόμη και από το SJF. Όμως γενικά μπορούμε να πούμε ότι ο αλγόριθμος έχει ταυτόσημη συμπεριφορά με τον SJF για τους κόμβους 2 και 3. Το μόνο, ίσως, σημείο της γραφικής που ίσως φανεί παράξενο είναι η αύξηση της καθυστέρησης για $C = [1.0\ 5.5\ 1.5]$ αν και ο συνολικός ρυθμός εξυπηρέτησης του συστήματος αυξάνεται από το προηγούμενο πείραμα. Στην πραγματικότητα, όμως, ο ρυθμός εξυπηρέτησης που βλέπει ο κόμβος 2 στις ουρές των άλλων δύο μειώνεται αφού από $1+2.5=3.5$ γίνεται $1+1.5=2.5$. Έτσι, όπως είναι φυσιολογικό η καθυστέρηση θα ανέβει και θα γίνει περίπου ίση με αυτή του πειράματος με $C = [1.0\ 4\ 1.5]$ όπου πάλι ο 2 βλέπει τον ίδιο συνολικό ρυθμό εξυπηρέτησης.

Τέλος, κύριο χαρακτηριστικό του παρακάτω γραφήματος είναι συνεχώς μειούμενη διαφορά μεταξύ τω ν αλγορίθμων όσο αυξάνεται ο συνολικό ρυθμός εξυπηρέτησης του συστήματος, κάτι που είναι αναμενόμενο αφού το σύστημα είναι όλο και λιγότερο φορτωμένο και επομένως όλοι οι αλγόριθμοι καταφέρνουν να εξυπηρετήσουν ικανοποιητικά τις αιτήσεις των κόμβων.

Στην επόμενη σελίδα, φαίνεται το διάγραμμα καθυστερήσεων για κάθε κόμβο ξεχωριστά.



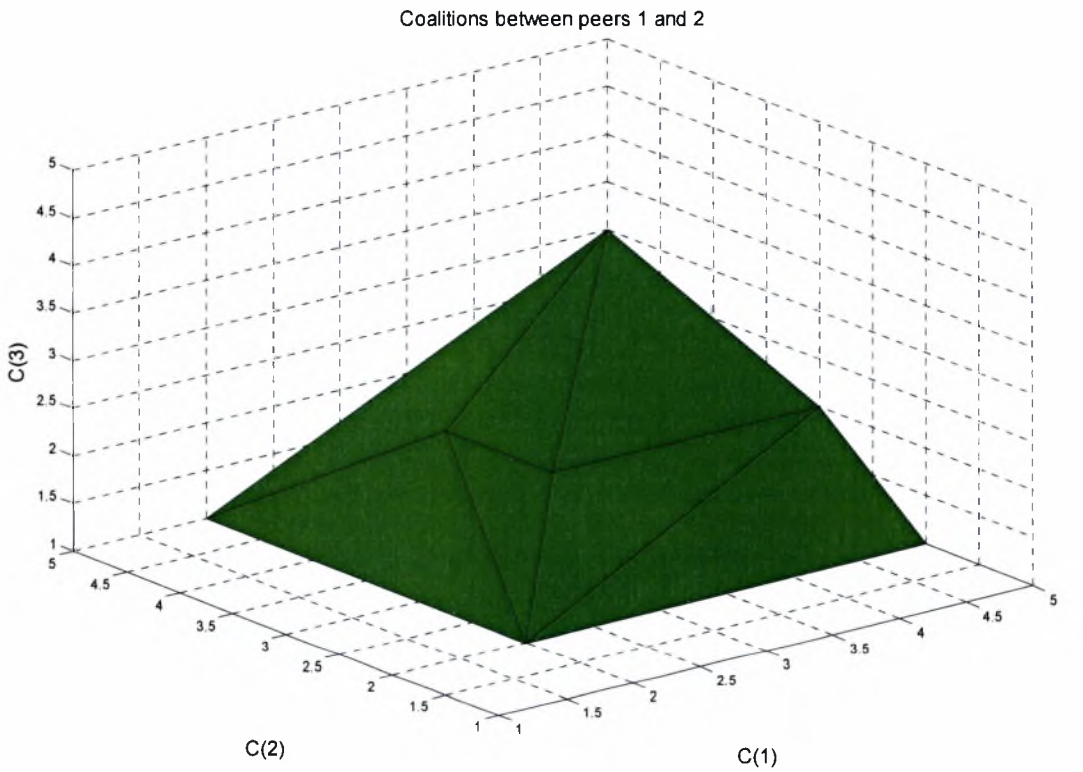
Όμως για να έχει κανείς μια καλή εικόνα για την απόδοση ενός αλγορίθμου είναι απαραίτητη η γνώση της συμπεριφοράς του σε όλο το σύστημα συνολικά. Για το λόγο αυτό παραθέτουμε στη συνέχεια και το διάγραμμα της συνολικής μέσης καθυστέρησης στο σύστημα.



Όπως παρατηρούμε στο γράφημα, ο FCFS δίνει τη μεγαλύτερη συνολική καθυστέρηση στο σύστημα. Ο αλγόριθμος δυναμικών προτεραιοτήτων έχει λίγο καλύτερα αποτελέσματα από τον FCFS. Στην περίπτωση αυτή, παρατηρούμε ότι αν και για τους κόμβους που σχηματίζουν τη συμμαχία η απόδοση είναι ταυτόσημη με αυτή του SJF, η συνολική εικόνα του δεν είναι η ίδια. Εξαιτίας της μεγάλης αύξησης στην καθυστέρηση του κόμβου 1, που δε συμμετέχει στη συμμαχία, ο μέσος όρος του συστήματος ανεβαίνει αρκετά πάνω από τον αλγόριθμο SJF, ο οποίος είναι όπως αναμενόταν ο καλύτερος. Τέλος, επιβεβαιώνουμε τη συνεχώς φθίνουσα πορεία του μέσου χρόνου ανάκτησης εξαιτίας του συνεχώς αυξανόμενου συνολικού ρυθμού εξυπηρέτησης του συστήματος αλλά και το γεγονός ότι οι αλγόριθμοι συγκλίνουν σταδιακά, γεγονός που οφείλεται, όπως είπαμε και προηγουμένως, στο συνεχώς μειούμενο φόρτο του συστήματος και στη μικρή διαφορά απόδοσης των αλγορίθμων στις περιπτώσεις αυτές.

III. Συμμαχία κόμβων (Coalition)

Ένα από τα κύρια χαρακτηριστικά του αλγορίθμου είναι η πιθανότητα εμφάνισης *συμμαχιών (coalitions)* μεταξύ των κόμβων με το μεγαλύτερο ρυθμό εξυπηρέτησης. Θυμίζουμε ότι με τον όρο *Συμμαχία (Coalition)* εννοούμε το φαινόμενο κατά το οποίο δύο κόμβοι δίνουν αμοιβαία προτεραιότητα στην εξυπηρέτηση των αιτημάτων τους σε βάρος της εξυπηρέτησης του τρίτου κόμβου του δικτύου. Για να ανιχνεύσουμε τις περιπτώσεις όπου εμφανίζεται συμμαχία μεταξύ δύο κόμβων για ένα συγκεκριμένο διάνυσμα ρυθμών εξυπηρέτησης `C`, ελέγχουμε τον πίνακα `reputation`. Όταν η τιμή φήμης που έχει ο ένας για τον άλλο ξεπερνά ένα κατώφλι, έστω t_1, και αυτό συμβαίνει για κάποιο μεγάλο ποσοστό των εποχών, έστω t_2, τότε θεωρούμε ότι για το συγκεκριμένο διάνυσμα `C` αναπτύχθηκε μια συμμαχία. Στο επόμενο διάγραμμα φαίνεται η περιοχή όπου εμφανίζεται coalition μεταξύ των κόμβων 1 και 2 για τιμές των ρυθμών εξυπηρέτησης εντός του διαστήματος [1,5]. Σημειώνουμε ότι η προσομοίωση έγινε για ρυθμό άφιξης κάθε κόμβου ίσο με $\lambda = 0.9$ ενώ τα κατώφλια ορίστηκαν σε $t_1 = 0.6$ και $t_2 = 0.7$.



Το διάγραμμα αυτό μας δίνει πολλά στοιχεία για τις περιπτώσεις όπου εμφανίζονται coalition μεταξύ δύο κόμβων. Αρχικά, παρατηρούμε ότι κύριο χαρακτηριστικό του χώρου αυτού είναι ότι οι τιμές των ρυθμών εξυπηρέτησης των κόμβων 1 και 2 είναι και οι δύο μεγαλύτερες από αυτόν του κόμβου 3. Έτσι, λοιπόν, βλέπουμε ότι, εξαιτίας του τρόπου που αποφασίζονται οι προτεραιότητας και τα σπασίματα στον αλγόριθμο, οι κόμβοι 1 και 2 αντιλαμβάνονται ότι ο κόμβος 3 είναι πολύ αργότερος από αυτούς και έτσι δεν έχουν κανένα κέρδος αν δώσουν σε αυτόν

προτεραιότητα. Αντίθετα, ο κόμβος 2 βλέπει ότι αν δώσει προτεραιότητα στον κόμβο 1, θα ωθήσει και τον 1 στο να δίνει και εκείνος προτεραιότητα στα δικά του αιτήματα με αποτέλεσμα να μειώνεται κατά πολύ η μέση καθυστέρησή του. Όσο όμως αυξάνεται ο ρυθμός εξυπηρέτησης του κόμβου 3, τα σημεία όπου έχουμε συμμαχία μεταξύ των δύο πρώτων κόμβων μειώνονται και μετά από την τιμή $C(3)=3$ παύουν να δημιουργούνται. Αυτό συμβαίνει γιατί πλέον ο κόμβος 3 έχει αρκετά μεγάλο ρυθμό εξυπηρέτησης ώστε να μη συμφέρει τους κόμβους 1 και 2 να συνεργαστούν. Αυτό επιβεβαιώνεται και από το γεγονός ότι η τιμή φήμης μεταξύ 1 και 2 πέφτει κάτω από το κατώφλι 0.6 το οποίο είχαμε ορίσει.

Τέλος, όπως είναι φυσιολογικό αντίστοιχα διαγράμματα προκύπτουν και για τις περιπτώσεις που δημιουργούνται coalition μεταξύ των 1 και 3 ή μεταξύ των κόμβων 2 και 3. Όμως, όπως είναι φυσιολογικό ο χώρος που εμφανίζονται αυτές θα βρίσκεται σε άλλη θέση του τρισδιάστατου χώρου που δημιουργείται για διάφορες τιμές των $C(1), C(2)$ και $C(3)$.

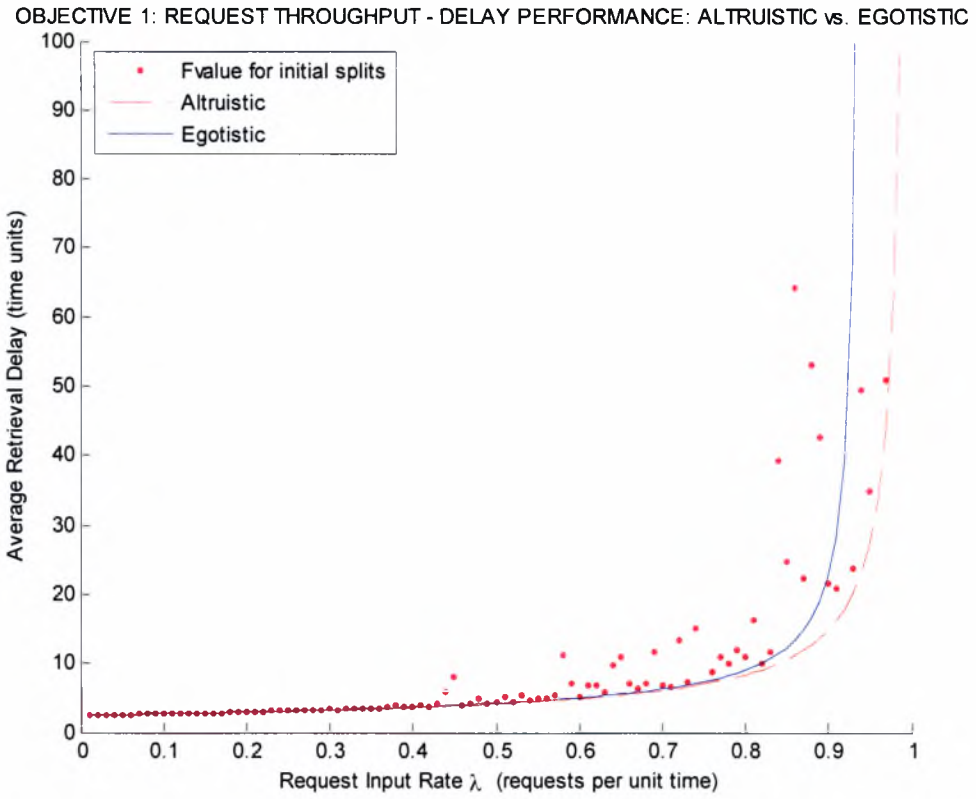
Συγκρίσεις αλγορίθμων

I. Σύγκριση Απόδοσης αλγορίθμων

Στο εδάφιο αυτό θα συγκρίνουμε την αλτρουιστική προσέγγιση επίλυσης του προβλήματος ελαχιστοποίησης του μέσου χρόνου ανάκτησης ενός αρχείου με τους αλγορίθμους της εγωιστικής προσέγγισης, που περιγράφονται αναλυτικά στο [15].

Έτσι, για τη σύγκριση με την αλτρουιστική προσέγγιση θα χρησιμοποιηθεί το πρωτόκολλο 1 της εγωιστική προσέγγισης. Ακόμη, στη ίδια σύγκριση θα χρησιμοποιούμε τον αλτρουιστικό αλγόριθμο που βασίζεται στη μέθοδο Waterfilling. Ο λόγος είναι ότι θεωρούμε πως λόγω της μικρής πολυπλοκότητάς του μπορεί να χρησιμοποιηθεί σε κάποιες εφαρμογές όπου ο Subglobal αλτρουιστικός αλγόριθμος δεν μπορεί γεγονός που καθιστά πιο ενδιαφέρουσα τη συμμετοχή του στη σύγκριση.

Στο παρακάτω γράφημα φαίνονται τα συνολικά βεβαρημένα αθροίσματα των δύο αλγορίθμων για την περίπτωση του δικτύου των τεσσάρων κόμβων και για ρυθμούς αφίξεων από 0.01 μέχρι 0.99 καθώς και οι καθυστερήσεις για την αρχική λύση από την οποία ξεκίνησαν οι αλγόριθμοι.



Συγκρίνοντας τις καθυστερήσεις για τις τυχαίες αρχικές τιμές με τις τελικές τιμές των αλγορίθμων, βλέπουμε ότι για μικρά λ_i δεν έχουν μεγάλη διαφορά από τα αποτελέσματα των δύο αλγορίθμων, αν και πάντα είναι χειρότερα και από τους δύο. Αυτό συμβαίνει γιατί σε ένα σύστημα με μικρό φόρτο εκ των πραγμάτων οι καθυστερήσεις θα είναι μικρές. Αντίθετα, για μεγάλες τιμές φόρτου βλέπουμε ότι και οι δύο αλγόριθμοι βελτιώνουν σημαντικά την απόδοση του συστήματος.

Επίσης, όπως εξηγείται και στο [15], ο εγωιστικός αλγόριθμος δεν καταφέρνει να βρει ένα εφικτό διάνυσμα για πολύ μεγάλους ρυθμούς αφίξεων και για αυτό το λόγο η γραφική του εγωιστικού αλγορίθμου δεν έχει τιμές για λ_i μεγαλύτερο από 0.94. Επίσης, δεν απεικονίζονται στο γράφημα τιμές καθυστέρησης μεγαλύτερες από 100, που αντιστοιχούν σε μεγάλα λ_i, για να είναι πιο ευδιάκριτες οι διαφορές των αλγορίθμων στις υπόλοιπες τιμές. Ενδεικτικά αναφέρουμε τις παρακάτω τιμές για κάποια από τα λ_i της προσομοίωσης.

λ_i / Αλγόριθμος	**0.5**	**0.7**	**0.8**	**0.9**	**0.94**
Εγωιστικός	4.242	6.342	9.083	22.55	432.1
Αλτρουιστικός	4.209	6.163	8.424	14.82	23.16

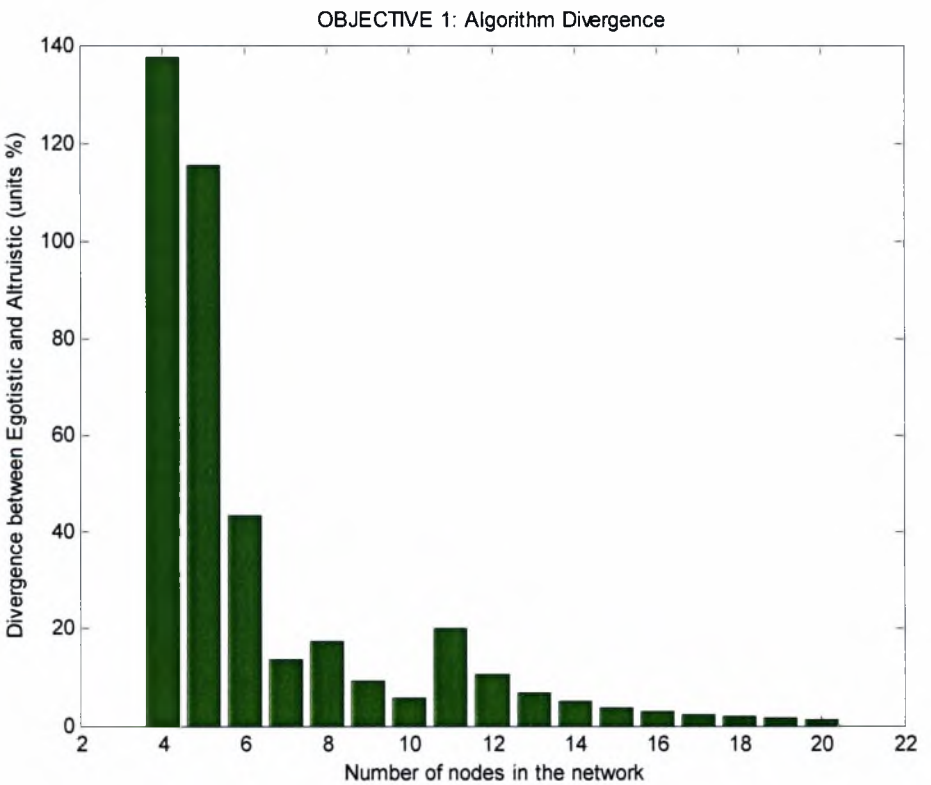
Γενικά, παρατηρούμε ότι για μικρές τιμές λ_i οι καθυστερήσεις που μας δίνουν οι αλγόριθμοι δεν απέχουν πολύ, ενώ για τιμές πάνω από το 0.8 ο εγωιστικός αλγόριθμος αρχίζει να μας δίνει σημαντικά χειρότερα αποτελέσματα. Αυτή η συμπεριφορά κορυφώνεται για $\lambda_i = 0.94$ όπου ο εγωιστικός αλγόριθμος καταφέρνει

μετά βίας να βρει ένα εφικτό διάνυσμα, με πολύ μεγάλη όμως καθυστέρηση. Ο λόγος που συμβαίνει αυτό είναι ότι όσο το σύστημα λειτουργεί με μικρό φόρτο η λήψη αποφάσεων με βάση καθαρά εγωιστικά κριτήρια αποδεικνύεται μια σχεδόν βέλτιστη στρατηγική. Καθώς όμως τα λ_i αρχίζουν να μεγαλώνουν, η εγωιστική στρατηγική δεν έχει το ίδιο καλά αποτελέσματα για το συνολικό δίκτυο. Στις περιπτώσεις αυτές οι επιλογές των κόμβων με μεγάλη προτεραιότητα περιορίζουν πολύ τις επιλογές των υπολοίπων, οι οποίοι αδυνατούν να σπάσουν τα αιτήματά τους με τον τρόπο θα ήθελαν και αναγκάζονται να καταφύγουν σε λύσεις που, αν και είναι οι καλύτερες δυνατές με τις συγκεκριμένες συνθήκες, αυξάνουν πολύ τη συνολική καθυστέρηση.

Στη συνέχεια, παραθέτουμε ένα γράφημα με την ποσοστιαία διαφορά απόδοσης των δύο αλγορίθμων για διαφορετικούς αριθμούς κόμβων στο δίκτυο. Για κάθε αριθμό κόμβων τρέξαμε την προσομοίωση για όλα τα λ_i εντός του διαστήματος [0.8, 0.99] εκτός από εκείνα όπου ο εγωιστικός αλγόριθμος αδυνατούσε να λύσει το πρόβλημα. Για κάθε διαφορετική τιμή λ_i, υπολογίζαμε το ποσοστό βελτίωσης που επιφέρει ο αλτρουιστικός αλγόριθμος επί του αποτελέσματος του εγωιστικού, δηλαδή το ποσοστό που δηλώνει ο τύπος:

$$ratio = \frac{egotistic - altruistic}{altruistic} \qquad (4.12)$$

Στη συνέχεια, υπολογίζουμε το μέσο όρο για όλες τις τιμές των λ_i για ένα συγκεκριμένο αριθμό κόμβων και τα παρουσιάζουμε με τη μορφή ράβδων.



Βλέποντας το παραπάνω γράφημα μπορούμε να πούμε γενικά ότι υπάρχει μια σαφής τάση μείωσης της διαφοράς των δύο αλγορίθμων. Αρχικά, για το δίκτυο των

τεσσάρων κόμβων βλέπουμε ότι ο εγωιστικός αλγόριθμος αποδίδει περίπου 140% χειρότερα από τον αλτρουιστικό. Αυτό συμβαίνει εξαιτίας της κακής συμπεριφοράς του αλγορίθμου στα μεγάλα λ_i, όπως εξηγήσαμε και παραπάνω. Στη συνέχεια όμως, όσο αυξάνεται ο αριθμός των κόμβων ο αλγόριθμος καταφέρνει να λύσει καλύτερα το πρόβλημα ελαχιστοποίησης με αποτέλεσμα να πλησιάζει, για μεγάλο αριθμό κόμβων, την απόδοση του αλτρουιστικού. Μοναδική παραφωνία στη συνεχόμενη μείωση της διαφοράς στην απόδοση αποτελούν οι περιπτώσεις των οκτώ και των έντεκα κόμβων στο σύστημα. Αυτό όμως δεν αλλάζει τη γενική εικόνα αλλά οφείλεται στον παρακάτω λόγο. Η προσομοίωση για δέκα κόμβους, για παράδειγμα, σταματά στην τιμή $\lambda_i = 0.98$, γιατί από το σημείο αυτό και πέρα ο εγωιστικός αλγόριθμος δεν μπορεί να βρει ένα εφικτό διάνυσμα λύσης. Όμως, στην προσομοίωση των έντεκα κόμβων, ο εγωιστικός αλγόριθμος καταφέρνει να λύσει και την περίπτωση που $\lambda_i = 0.99$. Επομένως στον μέσο όρο περιλαμβάνεται και η διαφορά των αποδόσεων για την τιμή $\lambda_i = 0.99$. Όμως, ο αλγόριθμος σε αυτό το λ_i καταφέρνει απλώς να βρει μια εφικτή λύση που όμως απέχει πολύ από τη βέλτιστη λύση του αλτρουιστικού αλγορίθμου ως προς το συνολικό βεβαρημένο άθροισμα. Αυτό έχει ως αποτέλεσμα, η διαφορά αυτή να αυξήσει πολύ το μέσο όρο. Στον ίδιο λόγο οφείλεται και η μικρή αύξηση της διαφοράς και στην περίπτωση των οκτώ κόμβων.

Εκ πρώτης όψεως, το γεγονός της μείωσης της διαφοράς των αλγορίθμων καθώς αυξάνεται ο αριθμός των κόμβων φαίνεται παράξενο. Όμως, αν σκεφτούμε ότι το πρόβλημα που προσπαθούμε να λύσουμε αποτελεί ένα πρόβλημα *Εξισορρόπησης Φόρτου (Load balancing)* και προσπαθήσουμε να το εξηγήσουμε σαν τέτοιο, θα δούμε αμέσως ότι το συμπέρασμα είναι απολύτως λογικό. Υποθέτουμε ότι έχουμε ένα δίκτυο τεσσάρων κόμβων και κάθε κόμβος παράγει αιτήματα με ρυθμό $\lambda_i = 0.6$. Στη συνέχεια, θα προσπαθήσουμε να εξηγήσουμε τη συμπεριφορά των δύο πρώτων κόμβων. Αρχικά, ο κόμβος 1, όπως είναι φυσιολογικό θα μοιράσει το ρυθμό του ισόποσα στους τρεις άλλους κόμβους του δικτύου, και έτσι προκύπτει η πρώτη σειρά του επόμενου πίνακα. Ο κόμβος 2, πρέπει να μοιράσει έτσι τα λ του ώστε να έχει την ίδια καθυστέρηση παντού, που σημαίνει ότι θα πρέπει κάθε άλλος κόμβος να έχει συνολικά τον ίδιο ρυθμό αφίξεων. Για αυτό, βλέποντας ότι οι κόμβοι 3 και 4 έχουν ήδη ρυθμό άφιξης 0.2 ενώ ο 1 δεν έχει καθόλου, αποφασίζει να στείλει 0.2 στον 1. Στη συνέχεια, όπως είναι φυσιολογικό αποφασίζει να μοιράσει ισόποσα το υπόλοιπο 0.4, αφού πλέον όλοι οι άλλοι κόμβοι δέχονται αιτήματα με τον ίδιο ρυθμό. Επομένως, στέλνει από $\frac{0.4}{N-1} = 0.1333$ σε κάθε ένα. Κάθε κόμβος ακολουθώντας αυτή την εγωιστική λογική αποφασίζει ποια θα είναι τα σπασίματά του ώστε να μειώσει την δική του καθυστέρηση. Έτσι, αμέσως μετά τη λήψη μιας τέτοιας απόφασης από τον καθένα, οι συνολικοί ρυθμοί αφίξεων των άλλων κόμβων είναι ίσοι μεταξύ τους αλλά όχι και ίσοι και με το δικό του συνολικό ρυθμό αφίξεων, όπως θα έπρεπε να συμβαίνει στη βέλτιστη λύση για το σύστημα. Ο λόγος είναι ότι κάθε κόμβος δεν μπορεί να στείλει αιτήματα στον εαυτό του με αποτέλεσμα ο ίδιος να έχει λιγότερο συνολικό φόρτο από όσο θα έπρεπε. Για τον λόγο αυτό στο τέλος του αλγορίθμου ο τελευταίος κόμβος θα δέχεται συνολικά μικρότερο ρυθμό αιτημάτων από ότι όλοι οι άλλοι και αυτή η διαφορά είναι που αυξάνει το συνολικό βεβαρημένο άθροισμα των καθυστερήσεων έναντι του αλτρουιστικού αλγορίθμου. Ενδεικτικά, αναφέρουμε ότι στο τέλος της εκτέλεσης του αλγορίθμου ο πρώτος κόμβος δέχεται συνολικό ρυθμό 0.6366, ο δεύτερος και ο τρίτος δέχονται 0.6369, ενώ ο τέταρτος 0.4888. Στον παρακάτω πίνακα φαίνονται αναλυτικά οι επιλογές των κόμβων με βάση τον εγωιστικό αλγόριθμο.

Κόμβος 1	Κόμβος 2	Κόμβος 3	Κόμβος 4
0	$\frac{0.6}{N-1}=0.2$	$\frac{0.6}{N-1}=0.2$	$\frac{0.6}{N-1}=0.2$
$\frac{0.6}{N-1}+\frac{0.4}{N-1}=0.3333$	0	$\frac{0.4}{N-1}=0.1333$	$\frac{0.4}{N-1}=0.1333$
$\frac{0.4667}{N-1}=0.1555$	$\frac{0.4}{N-1}+\frac{0.4667}{N-1}=0.2888$	0	$\frac{0.4667}{N-1}=0.1555$
$\frac{0.4445}{N-1}=0.1481$	$\frac{0.4445}{N-1}=0.1481$	$\frac{0.4667}{N-1}+\frac{0.4445}{N-1}=0.3036$	0

Από τις επιλογές των κόμβων φαίνεται ότι ο ρυθμός που θα στείλουν στους άλλους κόμβους για ένα δεδομένο συνολικό ρυθμό λ_i εξαρτάται από τον αριθμό των κόμβων του δικτύου N και μάλιστα είναι ποσότητες αντιστρόφως ανάλογες. Έτσι λοιπόν, μπορούμε να βγάλουμε εύκολα το συμπέρασμα ότι όσο πιο πολλοί είναι οι κόμβοι του δικτύου τόσο πιο μικρή θα είναι η διαφορά των συνολικών ρυθμών αφίξεων των κόμβων στο τέλος του αλγορίθμου και επομένως τόσο πιο λίγο θα υστερεί ο εγωιστικός αλγόριθμος από τον αλτρουιστικό, που καταφέρνει να εξισορροπήσει το σπάσιμο των ρυθμών αφίξεων πιο αποτελεσματικά.

Λαμβάνοντας, λοιπόν, υπόψη το γεγονός ότι η διαφορά των αλγορίθμων μειώνεται όσο αυξάνεται ο αριθμός των κόμβων του δικτύου μας οδηγείτε κανείς στο συμπέρασμα ότι, τελικά, η εγωιστική συμπεριφορά φαίνεται να είναι η βέλτιστη στρατηγική για το δίκτυο. Όμως, σε κάποιες περιπτώσεις, τα δίκτυα που δημιουργούνται δεν είναι μεγάλα με αποτέλεσμα ο αλτρουιστικός αλγόριθμος να είναι προτιμότερος. Τέτοιες περιπτώσεις μπορούμε να πούμε ότι εμφανίζονται όταν κανείς θέλει να κατεβάσει κάποιο μη δημοφιλές αρχείο, το οποίο υπάρχει σε πολύ μικρό αριθμό χρηστών, ή στην περίπτωση αρχικής διάδοσης ενός νέου αρχείου. Τότε θα δημιουργηθεί ένα μικρό δίκτυο όπου η απόδοση του αλτρουιστικού αλγορίθμου θα είναι σημαντικά καλύτερη. Η σημασία των περιπτώσεων αυτών δεν μπορεί να υποτιμηθεί καθώς σε αυτές είναι περισσότερο δύσκολη η διάδοση των αρχείων.

Ολοκληρώνοντας τη σύγκριση των δύο προσεγγίσεων, στον πίνακα που ακολουθεί παρουσιάζουμε τα βασικά χαρακτηριστικά τους.

Εγωιστικός αλγόριθμος	Αλτρουιστικός αλγόριθμος
Κάθε κόμβος ελαχιστοποιεί μόνο τη δική του καθυστέρηση αδιαφορώντας για τους κόμβους με μικρότερη προτεραιότητα	Κάθε κόμβος προσπαθεί να ελαχιστοποιήσει την καθυστέρηση που έχει ο ίδιος και οι κόμβοι με μικρότερη προτεραιότητα από αυτόν.
Μη επαναληπτικός	Επαναληπτικός
Υλοποιείται με τη μέθοδο Waterfilling	Υλοποιείται με τη μέθοδο Waterfilling
Δεν μπορεί να λύσει το πρόβλημα ελαχιστοποίησης για δίκτυο λίγων κόμβων με πολύ μεγάλο φόρτο	Καταφέρνει να λύσει με το βέλτιστο τρόπο κάθε περίπτωση του προβλήματος ελαχιστοποίησης
Για την επίλυση του προβλήματος από κάποιο κόμβο απαιτείται η ανταλλαγή μηνυμάτων που περιέχουν το άθροισμα των σπασιμάτων των κόμβων με μεγαλύτερη προτεραιότητα σε κάθε εξυπηρετητή.	Για την επίλυση του προβλήματος, απαιτούνται επιπλέον από τον εγωιστικό αλγόριθμο τα σπασίματα κάθε κόμβου μικρότερης προτεραιότητας στην προηγούμενη επανάληψη.
Συγκλίνει στον αλτρουιστικό αλγόριθμο καθώς αυξάνονται οι κόμβοι του δικτύου	Η διαφορά της απόδοσής του με τον εγωιστικό μειώνεται όσο αυξάνεται ο αριθμός των κόμβων στο δίκτυο.
Αποτελεί τη βέλτιστη στρατηγική σε δίκτυα με μεγάλο αριθμό κόμβων	Αποτελεί τη βέλτιστη στρατηγική σε δίκτυα με λίγους κόμβους

II. Σύγκριση Ευρεστικών και Βέλτιστων αλγορίθμων - Επίδραση εγωιστικής συμπεριφοράς

Στη συνέχεια θα αναλύσουμε και θα παρουσιάσουμε την επίδραση της εγωιστικής συμπεριφοράς που χαρακτηρίζει τους ευρεστικούς αλγορίθμους, τόσο κατά την επιλογή των σπασιμάτων τους, όσο και κατά την ανάθεση των προτεραιοτήτων για την περίπτωση του αλγορίθμου προτεραιοτήτων.

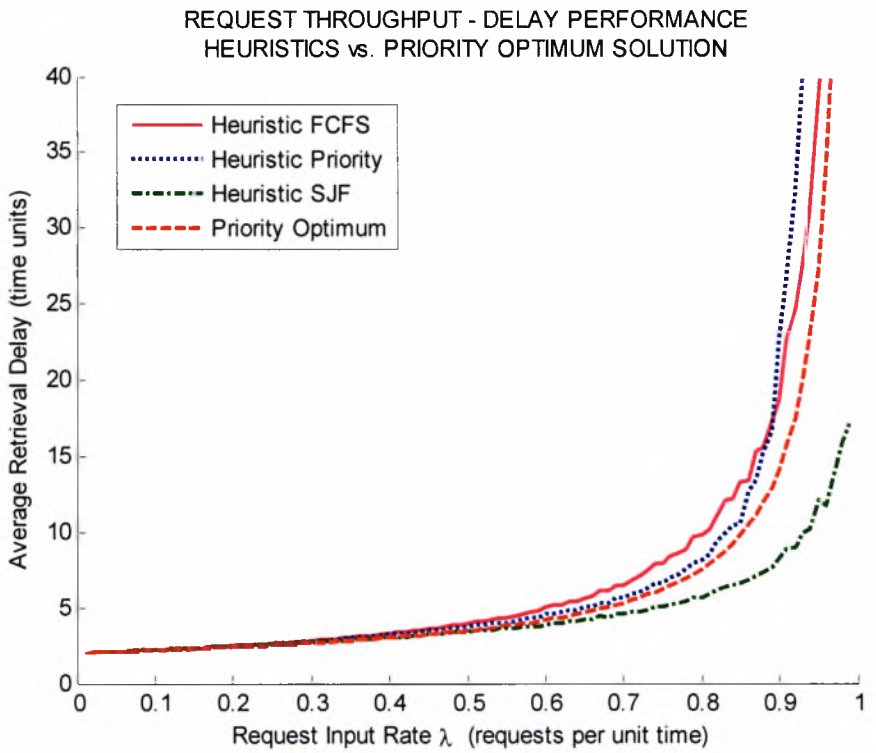
Για το σκοπό αυτό συγκρίνουμε την απόδοση των αλγορίθμων αυτών με τη βέλτιστη λύση που παρουσιάσαμε σε προηγούμενο εδάφιο. Ωστόσο, για να είναι δυνατή η σύγκριση θα χρησιμοποιήσουμε για όλες τις περιπτώσεις την βεβαρημένη μέση καθυστέρηση του συστήματος. Συγκεκριμένα η έκφραση που παρουσιάζουμε είναι η ακόλουθη

$$\sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} D_i = \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \sum_{j=1, j\neq i}^{N} \frac{\lambda_{ij}}{\lambda_i} D_{ij} \qquad (4.13)$$

Η σύγκριση αυτή θα ήταν άδικη αν διατηρούσαμε τον ευρεστικό αλγόριθμο προτεραιοτήτων ως έχει, καθώς αντιμετωπίζει όλους τους χρήστες ως ίδιας σημασίας για το σύστημα. Για να αντιμετωπίσουμε αυτό το φαινόμενο τροποποιήσαμε το κριτήριο με το οποίο ο κάθε εξυπηρετητής ορίζει τις προτεραιότητες. Ορίσαμε λοιπόν ο κάθε εξυπηρετητής να ζυγίζει την καθυστέρηση που αντιμετωπίζει σε κάθε κόμβο j αντιστρόφως ανάλογα με το βάρος του b_j. Με αυτό τον τρόπο, κάθε κόμβος τείνει

να δίνει πιο εύκολα προτεραιότητα στους κόμβους μεγαλύτερης βαρύτητας και έτσι επιτυγχάνουμε να ενσωματώσουμε και την έννοια των βαρών στον αλγόριθμων.

Στη συνέχεια παραθέτουμε το γράφημα που δείχνει την απόδοση των αλγορίθμων για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i, ο οποίος είναι κοινός για όλους τους κόμβους. Στη συγκεκριμένη προσομοίωση όλοι οι κόμβοι έχουν ρυθμό εξυπηρέτησης $\mu=1$ και το διάνυσμα βαρών είναι το **b**=[3 2 1].



Όπως παρατηρούμε από το παραπάνω γράφημα για μικρές τιμές λ_i όλοι οι αλγόριθμοι έχουν παρόμοια απόδοση, ενώ καθώς το λ_i αυξάνει, αυξάνεται και η διαφορά στην απόδοση τους. Όπως ήταν αναμενόμενο η SJF πολιτική είναι η καλύτερη αφού εκμεταλλεύεται την επιπλέον γνώση του χρόνου εξυπηρέτησης κάθε εργασίας. Επίσης παρατηρούμε ότι πάντοτε η βέλτιστη λύση είναι καλύτερη των άλλων δύο και ειδικά για μεγάλες τιμές φόρτου η διαφορά είναι σημαντική. Ενδεικτικά παραθέτουμε την καθυστέρηση για τους δύο αλγορίθμους προτεραιοτήτων στον παρακάτω πίνακα.

λ_i	0.6	0.7	0.8	0.9

Αλγόριθμος				
Ευρεστικός	4.565	5.705	8.195	22.76
Βέλτιστος	4.221	5.365	7.599	14.16

Η στρατηγική προτεραιοτήτων έχει ανάλογη απόδοση με την FCFS υπερτερώντας για μικρό φόρτο και υστερώντας όταν ο φόρτος στο σύστημα είναι ιδιαίτερα μεγάλος. Αυτό συμβαίνει γιατί οι δύο κόμβοι που δίνουν προτεραιότητα ο ένας στον άλλο εξασφαλίζουν μικρή καθυστέρηση για τους εαυτούς τους όμως ταυτόχρονα εκτινάσσουν την καθυστέρηση του τρίτου η οποία συμπαρασύρει και την συνολική καθυστέρηση του συστήματος.

Το συμπέρασμα λοιπόν στο οποίο καταλήγουμε είναι ότι η εγωιστική συμπεριφορά των κόμβων έχει ιδιαίτερα αρνητική επίδραση στη μέση καθυστέρηση του συστήματος, γεγονός που ενισχύεται καθώς αυξάνεται ο φόρτος στο σύστημα και καθώς εγκαθιδρύονται συνέργιες ανάμεσα σε κάποιους κόμβους.

Κεφάλαιο 5 – Ελαχιστοποίηση μέγιστου χρόνου ανάκτησης

Το πρόβλημα ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης βρίσκει εφαρμογή σε περιπτώσεις όπου ένας χρήστης στέλνει αιτήματα που αντιστοιχούν σε τμήματα κάποιας ολότητας, είτε αυτή είναι ένα αρχείο είτε κάποια άλλη εργασία που μπορεί να χωριστεί σε τμήματα. Όπως είναι κατανοητό, σε αυτή την περίπτωση δεν μας ενδιαφέρει ο μέσος χρόνος ανάκτησης αλλά ο μέγιστος από τους χρόνους εξυπηρέτησης αυτών των επιμέρους αιτημάτων. Έτσι λοιπόν, σε αντιστοιχία με όσα αναφέραμε για την επίλυση του προηγούμενου προβλήματος, το νέο πρόβλημα ελαχιστοποίησης αναφέρεται στην εύρεση ενός πίνακα «σπασιμάτων» Λ καθώς και ενός πίνακα προτεραιοτήτων Π για κάθε κόμβο ώστε να ελαχιστοποιείται το παρακάτω πρόβλημα:

$$\min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij} \qquad (5.1)$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} \lambda_{ij} = \lambda_i \qquad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} > 0 \qquad \forall i, j$$

Η αντικειμενική συνάρτηση του παραπάνω προβλήματος είναι κυρτή (convex), αφού αποτελεί το σημείο προς σημείο μέγιστο κυρτών συναρτήσεων. Όμως, σε αντίθεση με το προηγούμενο πρόβλημα, δεν πρόκειται για συνεχή αντικειμενική συνάρτηση. Αυτό την καθιστά μη παραγωγίσιμη γεγονός που καθιστά δύσκολη την επίλυσή του με τις μεθόδους που χρησιμοποιήσαμε προηγουμένως. Ακόμη, δεν έχει αποδειχθεί ακόμη ότι και για το συγκεκριμένο πρόβλημα ισχύει ο κανόνας μC με αποτέλεσμα να μην μπορούμε να πούμε με σιγουριά ότι οι προτεραιότητες κάθε κόμβου καθορίζονται εξολοκλήρου από τα βάρη b_i ούτε ότι κάθε κόμβος έχει την ίδια προτεραιότητα σε κάθε εξυπηρετητή. Αυτό έχει σαν αποτέλεσμα το πρόβλημα (5.1) να μην μπορεί να αναχθεί σε ένα απλούστερο πρόβλημα εύρεσης μόνο του πίνακα Λ όπως είχε συμβεί στην προηγούμενη περίπτωση.

Παρά το γεγονός ότι δεν γνωρίζουμε αν ισχύει ο κανόνας μC εμείς θα υποθέσουμε ότι οι προτεραιότητες είναι σταθερές και προκαθορισμένες, και έστω σύμφωνα με τα βάρη b_i. Επομένως, θεωρούμε ότι ο πίνακας προτεραιοτήτων Π είναι γνωστός και ο μόνος άγνωστος στο πρόβλημα ελαχιστοποίησης είναι ο πίνακας των «σπασιμάτων» Λ.

Το πρόβλημα αυτό για να λυθεί πρέπει να μετατραπεί στην κανονική μορφή επίλυσης ενός κυρτού προβλήματος ελαχιστοποίησης. Αυτό μπορεί να γίνει εισάγοντας νέες μεταβλητές, έστω τις t_i, ώστε να αντικαταστήσουν τη συνάρτηση μεγίστου. Θέτοντας, λοιπόν, $t_i = \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij}$ και προσθέτοντας ένα ακόμη περιορισμό, το πρόβλημα παίρνει τη μορφή:

$$\min_{\lambda_{ij}} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} t_i \qquad (5.2)$$

$$s.t. \quad \sum_j \lambda_{ij} = \lambda_i$$

$$\frac{\lambda_{ij}}{\lambda_i} D_{ij} - t_i \le 0 \qquad \forall i, j$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \mathrm{j}$$

$$\lambda_{ij} \ge 0 \qquad \forall i, j$$

με $D_{ij} = \dfrac{1}{\left(1 - \sum_{k=1}^{i} \lambda_{kj}\right)\left(1 - \sum_{k=1}^{i-1} \lambda_{kj}\right)}$.

Ένας τρόπος επίλυσης του προβλήματος αυτού είναι με τη χρήση των συνθηκών ΚΚΤ. Για να γίνει αυτό πρέπει αρχικά να υπολογίσουμε τη συνάρτηση Lagrange, η οποία θα είναι η:

$$L = \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} t_i + \sum_{i=1}^{N} \xi_i \left(\lambda_i - \sum_{j=1, j\neq i}^{N} \lambda_{ij} \right) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \nu_{ij} \left(\frac{\lambda_{ij}}{\lambda_i \left(1 - \sum_{k=1}^{i} \lambda_{kj}\right)\left(1 - \sum_{k=1}^{i-1} \lambda_{kj}\right)} - t_i \right)$$

$$+ \sum_{j=1}^{N} \phi_j \left(1 - \sum_{j=1}^{N} \lambda_{ij} \right) \qquad (5.3)$$

Στη σχέση (5.3) παρατηρούμε ότι οι άγνωστοι είναι τα λ_{ij}, τα t_i και οι ΚΚΤ μεταβλητές ξ_i, ν_{ij} και ϕ_j.

Έπειτα, παραγωγίζουμε τη συνάρτηση Lagrange ως προς όλους αυτούς τους αγνώστους και θέτοντας κάθε μια μερική παράγωγο ίση με το μηδέν παίρνουμε τις εξισώσεις του συστήματος που πρέπει να λύσουμε για να βρούμε τη βέλτιστη λύση.

Αλτρουιστική Προσέγγιση

Περιγραφή Μεθόδου

Η Αλτρουιστική προσέγγιση επίλυσης του προβλήματος ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης βασίζεται στην ιδέα ότι κάθε κόμβος λύνει το πρόβλημα ελαχιστοποίησης για τον εαυτό του καθώς και για όλους τους κόμβους που έχουν μικρότερη προτεραιότητα από αυτόν, όπως ακριβώς συνέβαινε και στην αντίστοιχη προσέγγιση του προηγούμενου προβλήματος ελαχιστοποίησης (Βλ. Κεφάλαιο 4). Με τον τρόπο αυτό, κάθε κόμβος λαμβάνει υπόψη του την επίδραση των αιτήσεών του στις καθυστερήσεις των κόμβων με μικρότερη προτεραιότητα.

Κάθε κόμβος i λύνει το παρακάτω πρόβλημα ελαχιστοποίησης:

$$\min_{\lambda_{ij}} b_i \frac{\lambda_i}{\lambda} \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij} + \sum_{k>i} b_k \frac{\lambda_k}{\lambda} \max_{j \neq k} \frac{\lambda_{kj}}{\lambda_k} D_{kj} \qquad (5.4)$$

έτσι ώστε να ικανοποιούνται οι περιορισμοί που ίσχυαν και προηγουμένως. Το πρόβλημα όμως της σχέσης (5.4) είναι πολύ δύσκολο να λυθεί αφού η αντικειμενική συνάρτηση δεν είναι συνεχής λόγω της ύπαρξης της συνάρτησης μεγίστου. Για να αντιμετωπίσουμε αυτή τη δυσκολία κάνουμε την αντικατάσταση: $t_i = \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij}$ και για να εξαλείψουμε τη συνάρτηση μεγίστου η αντικατάσταση συνοδεύεται από την ανισότητα: $t_i \geq \frac{\lambda_{ij}}{\lambda_i} D_{ij} \quad \forall i, j$. Έτσι λοιπόν, το πρόβλημα (5.4) παίρνει τη μορφή:

$$\min_{\lambda_{ij}} b_i \frac{\lambda_i}{\lambda} t_i + \sum_{k>i} b_k \frac{\lambda_k}{\lambda} t_k \qquad (5.5)$$

$$s.t. \quad \sum_j \lambda_{ij} = \lambda_i$$

$$\frac{\lambda_{ij}}{\lambda_i} D_{ij} - t_i \leq 0 \qquad \forall i, j$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall j$$

$$\lambda_{ij} \geq 0 \qquad \forall i, j$$

με $D_{ij} = \frac{1}{\left(1 - \sum_{k=1}^{i} \lambda_{kj}\right)\left(1 - \sum_{k=1}^{i-1} \lambda_{kj}\right)}$

Με τον τρόπο αυτό φέρνουμε το πρόβλημα στην κανονική μορφή (standard form) ενός κυρτού προβλήματος ελαχιστοποίησης.

Το παραπάνω πρόβλημα ελαχιστοποίησης μπορεί να λυθεί είτε χρησιμοποιώντας τις συνθήκες ΚΚΤ, και λύνοντας το σύστημα των εξισώσεων που προκύπτει ακολουθώντας τη μεθοδολογία που εξηγήσαμε για την επίλυση του καθολικού προβλήματος νωρίτερα, είτε με κάποια συνάρτηση επίλυσης προβλημάτων ελαχιστοποίησης του matlab, όπως η fmincon.

Προσομοίωση

Λόγω της εγγενούς δυσκολίας επίλυσης της αλτρουιστικής προσέγγισης του προβλήματος, δεν ήταν εφικτό να λυθεί το πρόβλημα ακολουθώντας την αλτρουιστική προσέγγιση χρησιμοποιώντας το matlab. Έτσι, δεν παραθέτουμε αποτελέσματα προσομοίωσης για την αλτρουιστική προσέγγιση αλλά σκοπεύουμε μία από τις μελλοντικές επεκτάσεις της παρούσας διπλωματικής εργασίας να είναι η εύρεση κάποιας αποδοτικής μεθόδου επίλυσης του αλτρουιστικού προβλήματος ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης.

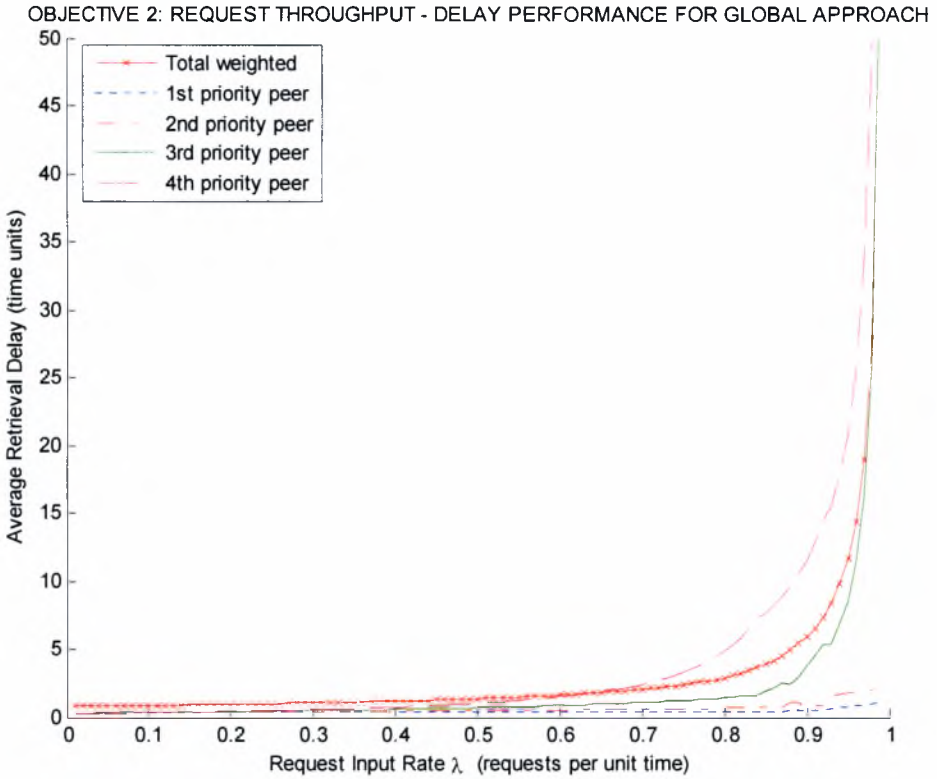
Καθολικό πρόβλημα

Για την καλύτερη κατανόηση της συμπεριφοράς και την αξιολόγηση της απόδοσης των μεθόδων επίλυσης του προβλήματος προχωρήσαμε στην προσομοίωση του καθολικού τρόπου επίλυσης χρησιμοποιώντας της συνθήκες ΚΚΤ. Η διαδικασία που ακολουθήθηκε ήταν ο υπολογισμός της συνάρτησης Lagrange και, στη συνέχεια, η χρήση των μερικών παραγώγων της, σε συνδυασμό με τις συνθήκες ΚΚΤ, ώστε να παραχθεί ένα σύστημα εξισώσεων από το οποίο θα προέκυπτε η βέλτιστη λύση. Η διαδικασία για τον υπολογισμό της συνάρτησης Lagrange (σχέση (5.3)) περιγράφεται στην αρχή του τρέχοντος κεφαλαίου. Το σύνολο των προσομοιώσεων πραγματοποιήθηκε σε περιβάλλον MATLAB και γενικά, εκτός αν αναφέρεται διαφορετικά, το δίκτυο που μοντελοποιούμε αποτελείται από N=4 κόμβους με διάνυσμα βαρών το **b**=[4 3 2 1]. Επίσης, όλοι οι κόμβοι έχουν τον ίδιο ρυθμό παραγωγής αιτήσεων λ_i.

Στη συνέχεια παρουσιάζουμε κάποια από τα αποτελέσματα που προέκυψαν.

Απόδοση αλγορίθμου

Στο παρακάτω γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός μέσος χρόνος ανάκτησης του συστήματος.



Ιδιαίτερο χαρακτηριστικό της γραφικής παράστασης είναι η μεγάλη αύξηση της καθυστέρησης καθώς αυξάνεται ο ρυθμός άφιξης αιτήσεων λ. Αυτό είναι απολύτως φυσιολογικό αφού ο φόρτος του συστήματος αυξάνεται καθώς μεγαλώνει το λ κάθε κόμβου, προκαλώντας μεγαλύτερη καθυστέρηση, και ιδιαίτερα όταν πλησιάζουμε στο ρυθμό εξυπηρέτησης C, όπου το σύστημα αγγίζει τα όρια ευστάθειάς του. Ένα άλλο σημαντικό χαρακτηριστικό του συστήματος, το οποίο έχουμε την ευκαιρία να επιβεβαιώσουμε με αυτή τη γραφική, είναι η πολύ μεγάλη καθυστέρηση που αντιμετωπίζει ο κόμβος 4, εκείνος δηλαδή με τη μικρότερη προτεραιότητα. Αυτό συμβαίνει γιατί ο αλγόριθμος δίνει πολύ μεγάλη βαρύτητα στην ελαχιστοποίηση του χρόνου ανάκτησης των κόμβων με μεγάλη προτεραιότητα, ιδιαίτερα του κόμβου 1, γεγονός που λειτουργεί αρνητικά για την καθυστέρηση των κόμβων με μικρότερη προτεραιότητα, και ιδιαίτερα του κόμβου 4. Άλλωστε, είναι εύκολο να αντιληφθούμε ότι ο κόμβος 1 έχοντας απόλυτη προτεραιότητα επηρεάζεται μόνο από την αύξηση του δικού του ρυθμού παραγωγής αιτήσεων, ενώ ο κόμβος 4 από την αύξηση όλων των λ_i. Ως επιβεβαίωση των παραπάνω, παρατηρούμε ότι οι κόμβοι με μεγάλη προτεραιότητα, δηλαδή οι 1 και 2, έχουν σχεδόν μηδενική καθυστέρηση (ο κόμβος 1 έχει, όπως ήταν φυσιολογικό, μικρότερη καθυστέρηση από τον 2) ενώ οι κόμβοι 3 και 4 έχουν αρκετά μεγαλύτερη. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.95$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα

D_1	D_2	D_3	D_4	Total Delay
0.7994	1.782	8.578	20.9	8.578

Τέλος, η συνολική καθυστέρηση στο σύστημα, όντας ο βεβαρημένος μέσος όρος των καθυστερήσεων με βάρη μεγαλύτερα της μονάδας, θα περιμέναμε να βρίσκεται πάνω από τις επιμέρους καθυστερήσεις των κόμβων. Αυτό γενικά ισχύει, όχι όμως και για μεγάλες τιμές του λ_i. Σε αυτές τις περιπτώσεις η τιμή της συνολικής καθυστέρησης είναι μικρότερη από αυτή του κόμβου 4 λόγω της ραγδαίας αύξησης που περιγράψαμε παραπάνω.

Σημείωση

Αρχικά υποθέσαμε ότι ο βέλτιστος τρόπος ανάθεσης των προτεραιοτήτων είναι εκ των προτέρων γνωστός και στη συνέχεια προχωρήσαμε στην εύρεση των βέλτιστων σπασιμάτων για την δεδομένη ανάθεση προτεραιοτήτων. Αυτό είναι αρκετό για την περίπτωση του Objective1 καθώς ο μc rule υποδεικνύει τη βέλτιστη ανάθεση προτεραιοτήτων. Ωστόσο κάτι τέτοιο δεν ισχύει για την περίπτωση του προβλήματος αυτού. Για παράδειγμα για $\boldsymbol{\lambda}$=[0.95 0.78 0.97 0.7] ο ορισμός των προτεραιοτήτων ως node1 > node4 > node2 > node3 δίνει καθυστέρηση 4.3162 ενώ η ακολουθία που προτείνει ο μc rule έχει καθυστέρηση 4.5733. Επομένως, καθώς δεν υπάρχει κάποιος προκαθορισμένος βέλτιστος τρόπος ανάθεσης προτεραιοτήτων απαιτείται η δημιουργία ενός ευρεστικού αλγορίθμου που σε συνδυασμό με τους αλγορίθμους εύρεσης των βέλτιστων σπασιμάτων θα βρίσκει την βέλτιστη λύση του προβλήματος.

Κεφάλαιο 6 – Μελλοντικές επεκτάσεις

Στην παρούσα εργασία μελετήσαμε την βεβαρημένη καθυστέρηση απόκτησης ενός αρχείου, η οποία εξαρτάται από τον διαχωρισμό των αιτήσεων στους κόμβους άλλα και από την πολιτική εξυπηρέτησης κάθε εξυπηρετητή. Θεωρήσαμε ότι οι παράμετροι b_i είναι σταθερές κατά τη διάρκεια της προσομοίωσης, γνωστές σε όλους τους χρήστες, και ότι εκφράζουν τη σημασία κάθε χρήστη για το δίκτυο. Επομένως, τα βάρη αυτά θα μπορούσαν να είναι οι καθολικές τιμές reputation κάθε χρήστη στο σύστημα, που καθορίζονται με κάποιον εκ των γνωστών αλγορίθμων διαχείρισης εμπιστοσύνης.

Ωστόσο, σε πραγματικά συστήματα, η καθολική τιμή reputation δεν είναι γνωστή στους χρήστες, αλλά ο καθένας έχει στη διάθεση του μια εκτίμηση των τιμών αυτών, που μεταβάλλεται με την πάροδο του χρόνου. Θα ήταν λοιπόν ιδιαίτερα ενδιαφέρον να εξετάσουμε την απόδοση των προτεινόμενων αλγορίθμων στην περίπτωση αυτή και την δυνατότητα ενσωμάτωσης αλγορίθμων διαχείρισης εμπιστοσύνης στο παρόν σύστημα.

Επίσης, κρίνεται απαραίτητη μια διεξοδικότερη εξέταση του προβλήματος ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης λόγω της σπουδαιότητας των πιθανών εφαρμογών του.

Τέλος, στο μέλλον θα ήταν δυνατή η μελέτη των παραπάνω προβλημάτων για άλλα μοντέλα ουρών ή και για άλλες εκφράσεις καθυστέρησης.

Βιβλιογραφία

[1] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel and D. D. Yao, "Optimal Peer Selection for P2P Downloading and Streaming".

[2] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer- to -Peer Networks".

[3] S. Sanghavi, B. Hajek and Laurent Massoulie, "Efficient Data Dissemination in Unstructured Networks".

[4] J. Sethuraman, M. Squillante, "Optimal Stochastic Scheduling in Multi-class Parallel Queues".

[5] S. Borst, "Stochastic Dynamic Programming & Control of Queues", http://www.cwi.nl/~sem/LNMB/2004/.

[6] J. Liang, R. Kumar, K. Ross, "The KaZaA Overlay: A Measurement Study".

[7] R. Kumar, K. Ross, "Peer-Assisted File Distribution: The Minimum Distribution Time"

[8] J. Mundinger, R. Weber and G. Weiss, "Optimal Scheduling of Peer to Peer File Dissemination"

[9] *J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips,* "The Bittorrent P2P File-Sharing System: Measurements and Analysis"

[10] D. Bertsimas, "The achievable region method in the optimal control of queueing systems; formulations, bounds and policies"

[11] B. Mortazavi and G. Kesidis, "A model of a reputation system for incentive engineering"

[12] R. Ma, S. Lee, John, C. Lui, D. Yau, "Incentive Resource Distribution in P2P Networks"

[13] K. Lai, M. Feldman, I. Stoica, J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks"

[14] L. Lai and H. El Gamal, "The Water-Filling Game in Fading Multiple Access Channels"

[15] Γκατζίκης Λάζαρος, "Διαχείριση και Διάδοση εμπιστοσύνης σε ομότιμα δίκτυα", Διπλωματική εργασία